Peramalan Debit Bendungan Dengan Menggunakan Metode *Backpropagation* dan Algoritme Genetika

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Beta Deniarrahman Hakim
NIM: 165150209111001

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

PENGESAHAN

PERAMALAN DEBIT BENDUNGAN DENGAN MENGGUNAKAN METODE *BACKPROPAGATION* DAN ALGORITME GENETIKA

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Beta Deniarrahman Hakim
NIM: 165150209111001

Skripsi ini telah diuji dan dinyatakan lulus pada
2 Agustus 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Indriati, S.T, M.Kom.	Dr.Eng. Ahmad Afif Supianto,S.Si,M.Kom
NIP: 19831013 201504 2 002	NIK: 2012018 20623 1 001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T.,M.T.,Ph.D.
NIP:19710518 200312 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 2 Agustus 2018

Beta Deniarrahman Hakim

NIM: 165150209111001

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Tuhan Yang Maha Esa, karena atas berkah, rahmat serta hidayah-Nya penulis mampu menyelesaikan penulisan skripsi Peramalan Debit Bendungan Menggunakan Algoritme Genetika dan Metode Backpropagation dengan baik.

Dalam pelaksanaan penelitian ini dan penulisan laporan ini penulis mendapat banyak bantuan dari berbagai pihak, baik secara moril maupun secara materil. Dalam kesempatan ini penulis ingin mengucapkan terima kasih kepada:

1. Pihak keluarga yang senantiasa memberikan do'a dan mendukung penulis sehingga Skripsi ini dapat terselesaikan.
2. Bapak Tri Astoto Kurniawan, S.T., M.T., Ph.D. selaku Ketua Jurusan Teknik Informatika Universitas Brawijaya Malang.
3. Ibu Indriati, S.T, M.Kom selaku dosen pembimbing I.
4. Bapak Dr. Eng. Ahmad Afif Supianto, S.Si, M.Kom selaku pembimbing II.
5. Seluruh pihak yang telah membantu kelancaran Praktik Kerja Lapangan yang tidak dapat kami sebutkan satu persatu.

Penulis menyadari bahwa dalam penyusunan laporan ini masih banyak kekurangan, sehingga saran dan kritik yang membangun sangat penulis harapkan. Akhir kata penulis berharap penelitian ini dapat membawa manfaat bagi semua pihak yang menggunakannya.

Malang, 2 Agustus 2018

Penulis

beta.hakim@gmail.com

ABSTRAK

Peramalan debit bendungan diperlukan untuk merencanakan rencana alokasi air untuk berbagai kebutuhan misalnya untuk Pembangkit Listrik Tenaga Air (PLTA), pengendalian banjir dan irigasi. Jaringan saraf tiruan dalam hal ini metode *backpropagation* memliki metode pembelajaran untuk mengubah bobot dari nilai arsitektur jaringan saraf tiruan tersebut. Algoritme genetika dapat melakukan optimasi bobot jaringan saraf tiruan untuk menghindari terjadinya lokal minimum yang menjadi kelemahan *backpropagation*. Algoritme genetika akan melakukan optimasi bobot jaringan saraf tiruan sehingga dihasilkan individu sebagai representasi bobot dengan nilai *fitness* terbaik bobot hasil dari proses optimasi dengan algoritme genetika kemudian digunakan sebagai bobot awal jaringan saraf tiruan metode *backpropagation*. Data yang digunakan sebagai data input adalah data time series debit bendungan bulan sebelumnya. Data yang digunakan adalah data debit bulanan mulai tahun 2008 sampai dengan tahun 2017. Data *input* akan diproses oleh sistem untuk menghasilkan nilai *output* yang merupakan nilai peramalan debit bendungan satu bulan ke depan. Parameter pelatihan optimal pelatihan algoritme genetika dan *backpropagation* adalah ukuran populasi=100, jumlah generasi=100, kombinasi Cr dan Mr adalah 0,6 dan 0,4, jumlah iterasi=500, nilai *learning rate*=0,7. Hasil pengujian menggunakan parameter optimal mendapatkan nilai MSE = 0,04188.

Kata kunci: *peramalan, debit bendungan, algoritme genetika, backpropagation*

ABSTRACT

Dam discharge forecasting is needed to plan water allocation plans for various needs such as for Hydropower plant), flood control and irrigation. Artificial neural network in this case backpropagation method has a learning method to change the weight of the value of the architecture of the artificial neural network. Genetic algorithms can optimize the weight of artificial neural networks to avoid the occurrence of a minimum local which is a weakness of backpropagation. Genetic algorithms will optimize the weight of the artificial neural network so individuals which are produced as a weight representation with the best fitness value resulting from the optimization process with the genetic algorithm then used as the initial weight of the artificial neural network backpropagation method. The data used as input data is the dam discharge time series data the previous months. The data used is monthly debit data from 2008 to 2017. Input data will be processed to produce an output value which is the forecasted value of the dam discharge in the next month. The optimal training parameters for genetic algorithm and backpropagation training are the population size=100, the generation=100, Cr and Mr combination 0,6 and 0,4, the number of iterations = 500, the value of learning rate = 0.7. The test results using optimal parameters get the MSE value = 0.04188.

Keywords: forecasting, dam discharge, genetic algorithm, backpropagation

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1 PENDAHULUAN

1.1 Latar belakang

Bendungan merupakan salah satu infrastruktur dalam pengelolaan sumber daya air. Sebagai bangunan tempat tampungan air, beberapa bendungan memiliki beberapa potensi secara ekonomis, salah satunya sebagai Pembangkit Listrik Tenaga Air (PLTA). Salah satu bendungan yang memiliki fungsi PLTA ini adalah Bendungan Sutami. Bendungan tersebut terletak di Desa Karangkates, Kabupaten Malang. Bendungan Sutami memiliki fungsi sebagai pembangkit listrik dengan kapasitas 3 X 35 mW. Fungsi lain dari bendungan ini adalah sebagai pengendali banjir di wilayah hulu Sungai Brantas. Fungsi lain yang tidak kalah penting adalah untuk mendukung irigasi pada area persawahan sekitar 34.000 ha (Perum Jasa Tirta I, 2014), sehingga dapat dikatakan bendungan juga ikut serta memiliki perananan yang penting dalam mendukung ketahanan energi dan pangan nasional.

Untuk mengetahui kapasitas maksimal energi yang dihasilkan oleh pembangkit listrik pada periode tertentu di Bendungan Sutami, harus diketahui dulu berapa debit masuk air/*inflow* yang masuk ke dalam bendungan. Semakin besar debit aliran air yang masuk ke dalam bendungan, semakin besar pula potensi energi listrik yang dapat dihasilkan. Bila debit suatu bendungan dapat diramalkan, maka hasil dari proses tersebut dapat digunakan untuk melakuka perencanaan alokasi air untuk memaksimalkan energi listrik yang dihasilkan turbin PLTA. Selain itu peramalan debit bendungan akan sangat penting dalam merencanakan pola operasi pintu air dalam mengendalikan banjir dan memenuhi kebutuhan irigasi. Peramalan debit bendungan ini juga digunakan sebagai salah satu dasar penyusunan Rancangan Alokasi Air Tahunan yang dilakukan oleh Tim Koordinasi Pengelolaan sumber Daya Air (TKPSDA) di wilayah Jawa Timur. TKPSDA sendiri merupakan tim yang dibentuk oleh Kementerian Pekerjaan Umum dan Perumahan Rakyat yang bertugas untuk mengatur pengelolaan sumber daya air di suatu provinsi.

Jaringan saraf tiruan metode *backpropagation* telah digunakan dalam beberapa penelitian untuk melakukan peramalan, antara lain peramalan tinggi muka air sungai (Somlek, et al., 2016) yang menghasilkan nilai *mean square error* (MSE) sebesar 0,1139, peramalan ketinggian muka air waduk (Anindita & Laksono, 2016) dengan hasil *Root Mean Square Error* (RMSE) sebesar 9,2142. Penelitian lain dilakukan untuk meramalkan kenaikan inflasi di Indonesia (Sari, et al., 2016) yang membandingkan penggunakan jaringan saraf tiruan menggunakan *backpropagation* dan Sugeno *Fuzzy Interference System*. Penelitian tersebut menunjukkan kehandalan metode *backpropagation* untuk melakukan suatu peramalan. Meskipun memiliki kehandalan dalam implementasi peramalan, metode *backpropagation* juga memiliki kelemahan. Kelemahan metode *backpropagation* adalah jaringan saraf tiruan metode *backpropagation* seringkali terjebak pada kondisi lokal minimum (Liu & Liu, 2016).

Algoritme genetika adalah salah satu jenis algoritme evolusi. Algoritme genetika merupakan salah satu algoritme yang banyak digunakan untuk melakukan optimasi karena mampu memberikan solusi permasalah yang cukup kompleks (Mahmudy, 2015). Algoritme genetika dapat digunakan untuk melakukan optimasi pada pembobotan nilai jaringan saraf tiruan. Beberapa penelitian yang menggabungkan algoritme genetika dan jaringan saraf tiruan antara lain peramalan aktifitas lalu lintas (Haviluddin, 2015). Pada penelitian ini data yang digunakan adalah data *time series* sebagai data latih dan data uji. Penggunaan algoritme genetika dan jaringan saraf tiruan metode backpropagtation menunjukkan hasil dengan MSE yang lebih baik dibanding tanpa ada optimasi dengan algoritme enetika. Penelitian mengenai peramalan polusi udara (Esfandani, 2015) menunjukkan hasil RMSE yang lebih baik bila metode *backpropagation* dilakukan optimasi dengan algoritme genetika. Penelitian selanjutnya adalah mengenai prediksi kerusakan perangkat lunak (Jayaraj & Raman, 2016) menunjukkan jaringan saraf tiruan dengan pelatihan *backpropagation* dan optimasi bobot dengan algoritme genetika menghasilkan hasil peramalan yang lebih baik.

Mengacu pada beberapa penelitian sebelumnya, pada penelitian ini akan dilakukan peramalan debit Bendungan Sutami menggunakan jaringan saraf tiruan dengan metode *backpropagation* yang dioptimasi bobotnya menggunakan algoritme genetika. Optimasi jaringan saraf tiruan menggunakan algoritme genetika memungkinkan penggunaan data-data yang ada sebagai bahan untuk pembelajaran dan menghasilkan data keluaran berupa peramalan debit masuk Bendungan Sutami di periode yang akan datang.

1.2 Rumusan masalah

1. Bagaimana mengimplementasikan metode *backpropagation* dan algoritme genetika untuk peramalan debit bendungan?
2. Bagaimana hasil pengujian dari metode *backpropagation* dan algoritme genetika untuk peramalan debit bendungan?
3. Bagaimana perbandingan hasil peramalan menggunakan metode *backpropagation* sebelum dan sesudah dilakukan optimasi bobot menggunakan algoritme genetika untuk peramalan debit bendungan?

1.3 Tujuan

1. Menerapkan implementasi metode *backpropagation* dan algoritme genetika untuk peramalan debit bendungan.
2. Menguji tingkat akurasi implementasi metode *backpropagation* dan algoritme genetika untuk peramalan debit bendungan.
3. Mengetahui perbandingan metode *backpropagation* sebelum dan sesudah dilakukan optimasi menggunakan algoritme genetika untuk peramalan debit bendungan.

1.4 Manfaat

1. Mengetahui cara implementasi metode *backpropagation* dan algoritme genetika untuk peramalan debit bendungan.
2. Mengetahui tingkat akurasi implementasi metode *backpropagation* dan algoritme genetika untuk peramalan debit bendungan.
3. Mengetahui peramalan debit bendungan pada bulan berikutnya sebagai bahan pertimbangan perencanaan alokasi air di bendungan.

1.5 Batasan masalah

1. Studi kasus berada di Bendungan Sutami, Kabupaten Malang.
2. Data yang digunakan adalah data debit per bulan dari tahun 2008 sampai dengan tahun 2017

1.6 Sistematika pembahasan

Penyusunan tugas akhir ini menggunakan kerangka pembahasan sebagai berikut:

BAB 1 PENDAHULUAN

Bab ini berisi latar belakang, rumusan masalah, tujuan penelitian, manfaat penelitian, batasan masalah serta sistematika penulisan peramalan debit bendungan dengan menggunakan metode *backpropagation* dan algoritme genetika.

BAB 2 LANDASAN KEPUSTAKAAN

Bab ini membahas tentang teori-teori penunjang penelitian dan referensi terkait penelitian-penelitian yang dilakukan sebelumnya.

BAB 3 METODOLOGI

Bab ini membahas metode dan tahapan dalam peramalan debit bendungan dengan menggunakan metode *backpropagation* dan algoritme genetika.

BAB 4 PERANCANGAN

Bab ini memaoarkan langkah kerja dan metode yang diterapkan pada peramalan debit bendungan menggunakan metode *backpropagation* dan algoritme genetika serta perhitungan manual yang dilakukans erta dan skenario pengujian yang akan dilakukan.

BAB 5 IMPLEMENTASI

Bab ini membahas implementasi algoritme genetika dan *backpropagation* pada peramalan debit bendungan

BAB 6 PENGUJIAN DAN ANALISIS

Bab ini ini memaparkan proses pengujian yang dilakukan termasuk skenario pengujian dan analisis hasil dari peramalan debit bendungan

BAB 7 PENUTUP

Bab ini berisi kesimpulan penelitian yang telah dilakukan disertai dengan saran yang bias dikembangkan dalam penelitian berikutnya.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Kajian Pustaka

Beberapa penelitian yang telah dilakukan sebelumnya dapat dijadikan acuan pustaka dalam penelitian ini. Penelitian yang telah dilakukan dianalisis untuk mendapatkan beberapa hal penting yaitu parameter, metode yang digunakan serta hasil yang penarikan kesimpulan pada penelitian-penelitian tersebut. Hasil dari kajian pustaka yang dilakukan dapat dilihat pada Tabel 2.1.

Tabel 2.1 Kajian Pustaka

Penulis	Obyek	Metode	Hasil
(Haviluddin, 2015)	Peramalan aktifitas lalu lintas Input: Data historis lalu lintas	Optimasi jaringan saraf tiruan dengan algoritme genetika Proses: Algoritma genetika untuk melatih bobot jaringan saraf tiruan	Nilai MSE sebesar 0,000565 menunjukkan optimasi bobot oleh algoritme genetika memperoleh hasil yang lebih baik dibanding hanya menggunakan *backpropagation*
(Esfandani, 2015)	Peramalan polusi udara Input: Unsur-unsur kimia dalam udara -	Metode *backpropagation* dengan optimasi bobot dengan algoritme genetika Proses: Algoritma genetika untuk melatih bobot jaringan saraf tiruan	Hasil prediksi polusi udara memberikan nilai yang lebih baik dibandingkan menggunakan *backpropagation* saja
(Somlek, et al., 2016)	Peramalan tinggi muka air sungai Input: Historis tinggi muka air	Jaringan saraf tiruan dengan pembelajaran *backpropagation* Proses - *Feedforward* - *backpropagation*	Ketinggian air sungai dengan MSE sebesar 0,1139 dan akurasi 90,1218%
(Anindita & Laksono, 2016)	Peramalan tinggi muka air waduk Input: Data historis tinggi muka air	Jaringan Saraf Tiruan dengan pembelajaran *backpropagation* Proses: Algoritma genetika untuk melatih bobot jaringan saraf tiruan	Ketinggian air waduk dengan RMSE sebesar 9,2142.

(Sari, et al., 2016)	Peramalan kenaikan inflasi di Indonesia Input: Data historis inlasi	Jaringan Saraf Tiruan dengan pembelajaran *backpropagation* Proses - *Feedforward* - *backpropagation*	Akurasi dan *error* pada metode *backpropagation* lebih baik dibandingkan Sugeno FIS pada penelitian ini
(Jayaraj & Raman, 2016)	Klasifikasi cacat pada perangkat lunak Input: - Ukuran - Nomor modul - Kuantitas dari cacat pada software	*Multi-layer perceptron* dioptimasi dengan algoritme genetika Proses: Algoritma genetika untuk melatih bobot jaringan saraf tiruan	Klasifikasi, diuji dengan *precision*, *recall*. Optimasi dengan algoritme gentika juga menghindari terjadinya konvergensi dini

2.2 Peramalan

Kegiatan peramalan merupakan suatu proses perkiraan suatu variabel untuk masa datang berdasarkan data yang diperoleh dari masa lampau (Adwandha, 2017). Peramalan yang dilakukan secara kuantitatif dapat dilakukan menggunakan metode statistika dan matematika, sedangkan peramalan secara kualitatif didasarkan pendapat para pakar dan ahli. Peramalan bertujuan untuk memperkirakan sesuatu yang mendekati hasil yang sebenarnya. Untuk dapat melakukan suatu peramalan secara optimal diperlukan suatu metode yang tepat dan dapat dipertanggungjawabkan.

2.3 Time Series

Time series merupakan kumpulan dari hasil pengamatan berdasar urutan waktu maupun kumpulan peristiwa dalam selang waktu tertentu (Adwandha, 2017). Beberapa penelitian menggunakan data *time series* untuk menentukan pola dari suatu kumpulan data. Data yang digunakan pada penelitian ini adalah data *time series* yaitu data debit bulanan secara urut mulai bulan Januari 2008 sampai bulan Desember 2017.

2.4 Algoritme genetika

Algoritme genetika adalah salah satu contoh dari algoritme evolusi. Algoritme evolusi sendiri adalah suatu jenis teknik optimasi yang terinspirasi oleh proses pada evolusi biologi. Algoritme genetika banyak digunakan untuk menyelesaikan masalah optimasi dalam beberapa kasus yang memiliki ruang pencarian yang

cukup kompleks. Algoritme genetika diperkenalkan pertama kali oleh John Holland dan beberapa rekan di Universitas Michigan (Mahmudy, 2015).

Beberapa keunggulan algoritme genetika terkait permasalahan optimasi:

1. Algoritme genetika merupakan algoritme yang berbasis populasi, sehingga dapat diimplementasikan pada permasalahan optimasi yang relatif luas dan cukup kompleks.
2. Individu dalam suatu populasi dibagi ke dalam sub-populasi dan diproses oleh computer secara parallel sehingga waktu penyelesaian masalah dapat dilakukan dengan lebih singkat.
3. Algoritme genetika menghasilkan himpunan solusi optimal pada permasalahan dengan banyak objektif.
4. Algoritme genetika dapat menggunakan banyak variabel untuk memecahkan suatu masalah. Variabel tersebut dapat bersifat diskrit maupun kontinyu.
5. Algoritme genetika dapat dikombinasikan dengan algoritme yang lain dalam menyelesaikan suatu permasalahan, misalnya dengan *fuzzy* atau jaringan saraf tiruan.

Solusi permasalahan dalam algoritme genetika direpresentasikan sebagai *fitness*. Fungsi *fitness* digunakan untuk mengetahui kualitas suatu individu. Semakin tinggi nilai *fitness*-nya, semakin baik pula individu tersebut untuk dijadikan solusi suatu pemasalahan (Mahmudy, 2015). Proses yang dilakukan dalam algoritme genetika yaitu inisialisasi, reproduksi, evaluasi dan seleksi. Inisialisasi adalah proses pembangkitan individu secara acak yang mempunyai gen tertentu. Reproduksi adalah proses untuk menghasilkan keturunan (*offspring*) baru dari individu dalam suatu populasi. Evaluasi adalah proses untuk menghitung nilai *fitness* pada kromosom. Semakin tinggi nilai *fitness* dari suatu kromosom, semakin baik pula kromosom tersebut untuk dijadikan sebagai calon solusi. Seleksi adalah proses pemilihan individu terbaik pada suatu populasi dan *offspring* yang dipertahankan pada generasi selanjutnya (Mahmudy, 2015).

Kromosom dapat direpresentasikan dalam bentuk biner dan pengodean *real*. Representasi dalam bentuk biner memiliki kelemahan yaitu tidak dapat menjangkau beberapa titik solusi jika rentang solusi berada dalam daerah kontinyu, selain itu untuk melakukan perubahan biner ke bilangan decimal atau sebaliknya akan menambah waktu perhitungan. Untuk itu pada penelitian ini akan digunakan representasi kromosom dengan pengkodean bilangan *real*.

2.4.1 Inisialisasi

Proses inisialisasi adalah proses pembangkitan himpunan individu baru yang dilakukan secara acak. Masing-masing individu memiliki susunan kromosom tertentu yang mewakili sebuah solusi terkait permasalahan yang akan dipecahkan. Himpunan individu yang telah dibangkitkan diletakkan dalam sebuah penampungan yang disebut dengan populasi. Daya tampung suatu populasi

disebut dengan *popSize*. Pada Tabel 2.2 ditunjukkan contoh beberapa kumpulan individu dalam populasi.

Tabel 2.2 Populasi dengan 10 individu

P	Kromosom				*fitness*
	x_1	x_2	x_3	x_4	
P_1	0,3	3,4	4,6	2,2	10,5
P_2	4,6	2,1	0,1	1,3	8,1
P_3	0,1	3,8	0,1	2,5	6,5
P_4	3,8	0,1	2,5	1,6	8,0
P_5	5,0	1,5	1,1	2,1	9.7
P_6	2,3	4,8	1,7	0,1	8,9
P_7	3,2	2.8	4.9	4,1	15,0
P_8	1,6	2,5	3,3	5,5	12,9
P_9	6,4	1,6	2,3	0,2	10,5
P_{10}	1,9	5,3	5,5	2,4	15,1

Keterangan:

- $P_1 - P_{10}$: representasi induk (*parent*)
- $x_1 - x_4$: representasi kromosom
- *fitness* : nilai *fitness* untuk tiap individu

2.4.2 Reproduksi

Proses reproduksi dilakukan untuk menghasilkan suatu keturunan dari individu-individu dalam populasi. Individu-individu dalam populasi akan dipilih dan akan menghasilkan keturunan yang disebut *offspring.* Pada proses reproduksi ini terdapat 2 mekanisme untuk menghasilkan *offspring*, yaitu *crossover* dan mutasi. Pada mekanisme *crossover*, harus ditentukan terlebih dahulu nilai *crossover rate* (*cr*). *Crossover rate* mempunyai fungsi untuk menunjukkan perbandingan *offspring* yang dihasilkan dari mekanisme *crossover* terhadap *popSize* sehingga *offspring* yang dihasilkan dapat dituliskan:

$$offspring = cr \; x \; popSize \tag{2.1}$$

Salah satu metode yang dapat diterapkan pada mekanisme *crossover* adalah *one-cut point*. Misal ditentukan nilai *cr*=0,2 dan *popSize*=10, maka nilai *offspring* yang dihasilkan adalah 0,2 x 10 = 2. Sebagai contoh P_2 dan P_5 terpilih sebagai *parent*, maka akan menghasilkan 2 *offspring*, yaitu C_1 dan C_2.

P_1	[0,3	3,4	4,6	2,2]
P_5	[5,0	1,5	1,1	2,1]
C_1	[0,3	3,4	1,1	2,1]
C_2	[5,0	1,5	4,6	2,2]

Pada mekanisme mutasi, harus ditentukan dahulu nilai *mutation rate* (*mr*). *Mutation rate* mempunyai fungsi untuk menunjukkan perbandingan *offspring* yang dihasilkan dari mekanisme mutasi terhadap *popSize* sehingg *offspring* yang dihasilkan dapat dituliskan:

$$offspring = mr \; x \; popSize \tag{2.2}$$

Salah satu metode yang diterapakan dalam mekanisme mutasi adalah *reciprocal exchange*. Metode ini memilih dua posisi secara acak dan menukar nilai masing-masing posisi. Misal ditentukan nilai *mr*=0,1 dan *popSize*-10, maka nilai *offspring* yang dihasilkan adalah 0,1 x 10 = 1. Sebagai contoh bila dipilih P_6 sebagai *parent* dan ditentukan posisi 2 dan 3, maka akan dihasilkan C_3.

P_6	[2,3	4,8	1,7	0,1]
C_3	[2,3	1,7	4,8	0,1]

2.4.3 Evaluasi

Proses evaluasi dilakukan untuk menghasilkan nilai *fitness* untuk tiap kromosom. Dari proses reproduksi, diperoleh himpunan individu seperti pada Tabel 2.3.

Tabel 2.3 Himpunan Individu

P	Kromosom				*fitness*
	x_1	x_2	x_3	x_4	
P_1	0,3	3,4	4,6	2,2	10,5
P_2	4,6	2,1	0,1	1,3	8,1
P_3	0,1	3,8	0,1	2,5	6,5
P_4	3,8	0,1	2,5	1,6	8,0
P_5	5,0	1,5	1,1	2,1	9.7
P_6	2,3	4,8	1,7	0,1	8,9
P_7	3,2	2.8	4.9	4,1	15,0
P_8	1,6	2,5	3,3	5,5	12,9
P_9	6,4	1,6	2,3	0,2	10,5
P_{10}	1,9	5,3	5,5	2,4	15,1
C_1	0,3	3,4	1,1	2,1	6,9

C_2	5,0	1,5	4,6	2,2	13,3
C_3	2,3	1,7	4,8	0,1	8,9

2.4.4 Seleksi

Proses seleksi dilakukan untuk mendapatkan individu serta *offspring* dari populasi yang akan dipertahankan pada generasi berikutnya. Salah satu metode yang digunakan adalah *elitism selection*. Pada metode ini adalah mengumpulkan semua individu dan *offspring* kemudian diambil individu dan *offspring* dengan nilai *fitness* terbaik sejumlah *popSize* untuk dilanjutkan pada generasi selanjutnya. Dari Tabel 2.2 didapatkan himpunan individu dengan nilai *fitness* masing-masing, berikutnya akan diambil sejumlah *popSize* = 10 dengan nilai *fitness* terbaik, sehingga akan didapatkan Himpunan Individu baru seperti pada Tabel 2.4.

Tabel 2.4 Himpunan Individu Baru

P(t+1)	*Asal P(t)*	Kromosom				*fitness*
		x_1	x_2	x_3	x_4	
P_1	P_{10}	1,9	5,3	5,5	2,4	15,1
P_2	P_7	3,2	2.8	4.9	4,1	15,0
P_3	C_2	5,0	1,5	4,6	2,2	13,3
P_4	P_8	1,6	2,5	3,3	5,5	12,9
P_5	P_1	0,3	3,4	4,6	2,2	10,5
P_6	P_9	6,4	1,6	2,3	0,2	10,5
P_7	P_5	5,0	1,5	1,1	2,1	9.7
P_8	C_3	2,3	1,7	4,8	0,1	8,9
P_9	P_6	2,3	4,8	1,7	0,1	8,9
P_{10}	P_2	4,6	2,1	0,1	1,3	8,1

2.5 Perhitungan MSE

Hasil yang didapatkan dari suatu proses peramalan tidaklah selalu sesuai dengan keinginan atau kenyataan sehingga terjadi penyimpangan atau kesalahan. Kesalahan ini dapat disebabkan oleh ketidakmampuan model peramalan untuk mengenali unsur lain yang ada pada deret data. *Mean Square Error* (MSE) merupakan salah satu metode yang digunakan untuk menghitung akurasi hasil peramalan dengan cara menghitung perbedaan suatu data aktual dengan data hasil peramalan. Nilai MSE didapatkan dengan menggunakan Persamaan 2.3.

$$MSE = \frac{1}{k}\sum_{k=1}^{n}(t_k - y_k)^2 \qquad (2.3)$$

2.6 Perhitungan *fitness*

Nilai *fitness* pada algoritme genetika digunakan sebagai salah satu indikasi apakah suatu kromosom layak dijadikan sebagai solusi suatu permasalahan atau tidak. Semakin tinggi nilai *fitness* suatu kromosom, semakin mungkin kromosom tersebut merupakan solusi permasalahan yang paling mendekati optimal. Nilai *fitness* dapat dihasilkan dari Persamaan 2.4.

$$fitness = \frac{1}{MSE} \quad (2.4)$$

2.7 Jaringan Saraf Tiruan

Jaringan saraf tiruan diperkenalkan oleh Warren McCulloch dan Walter Pitts pada tahun 1943. Metode ini merupakan suatu metode perhitungan yang seolah-olah meniru cara kerja otak manusia. Hubungan antar neuron merupakan karakteristik cara kerja otak manusia. Masing-masing neuron terhubung dengan neuron lain melalui penghubung yang mempunyai nilai bobot. Hubungan ini disebut juga dengan arsitektur, sedangkan metode untuk menentukan nilai bobot penghubungnya disebut dengan pelatihan/*training*.

Neuron-neuron yang saling terhubung dan tersusun dalam satu kelompok disebut juga lapisan atau *layer*. Secara umum terdapat dua lapisan, yaitu *input* dan *output*. Seringkali terdapat pula satu lapisan yang berada di antara kedua lapisan tersebut yang disebut lapisan tersembunyi atau *hidden layer*.

Lapisan *input* memiliki *node-node* yang berupa nilai masukan. Banyaknya neuron pada lapisan ini bergantung pada banyaknya *input* yang akan digunakan dalam proses pelatihan. Lapisan tersembunyi merupakan lapisan yang tidak dapat diamati secara langsung. Dalam suatu penelitian lapisan tersembunyi dapat pula berjumlah lebih dari satu. Lapisan *output* bergantung pada *input* dan lapisan tersembunyi. Lapisan ini menampilkan hasil akhir keluaran dari sistem.

2.7.1 Arsitektur dan Pembelajaran

Jaringan saraf tiruan sering diklasifikasikan dengan *single layer* dan *multi layer*. Pada jaringan saraf tiruan *single layer*, *node* pada lapisan *input* akan terhubung secara langsung dengan unit *output*. Suatu jaringan saraf tiruan *multilayer* terdiri dari 1 atau lebih jaringan di antara lapisan *input* dan *output*. Jaringan *multi layer* dapat menyelesaikan masalah yang lebih rumit daripada jaringan *single layer*.

Selain arsitektur, metode pelatihan nilai bobot (pelatihan) merupakan pembeda karakteristik yang penting pada jaringan saraf tiruan. Jaringan saraf tiruan akan dibuat model jika hubungan antara *input* dan *output* sudah diketahui secara pasti. Proses mempelajari hubungan antara *input* dan *output* dilakukukan dengan metode pembelajaran. Macam-macam tipe metode pembelajaran menurut (Fausset, 1994) yaitu:

a. *Supervised*

Supervised learning merupakan suatu pembelajaran yang terawasi dimana jika *output* yang diharapkan telah diketahui sebelumnya. Biasanya pembelajaran ini dilakukan dengan menggunakan data yang telah ada. Pada metode ini, setiap pola yang diberikan kedalam jaringan saraf tiruan telah diketahui *output*-nya. Satu pola *input* akan diberikan ke satu neuron pada lapisan *input*. Pola ini akan dirambatkan di sepanjang jaringan saraf hingga sampai ke neuron pada lapisan *output*. Lapisan *output* ini akan membangkitkan pola *output* yang nantinya akan dicocokkan dengan pola *output* targetnya. Bila terjadi perbedaan antara pola *output* hasil pembelajaran dengan pola *output* target, maka akan muncul *error*. Apabila nilai *error* ini masih cukup besar, itu berarti masih perlu dilakukan pembelajaran yang lebih lanjut. Contoh algoritme jaringan saraf tiruan yang menggunakan metode *supervised learning* adalah Hebbian (hebb rule), *Perceptron*, Adaline, Boltzman, Hapfield, dan *backpropagation*

b. *Unsupervised*

Unsupervised learning merupakan pembelajan yang tidak terawasi dimana tidak memerlukan target *output*. Pada metode ini tidak dapat ditentukan hasil seperti apa yang diharapkan selama proses pembelajaran, nilai bobot yang disusun dalam proses r*ange* tertentu tergantung pada nilai *output* yang diberikan. Tujuan metode *unsupervised learning* ini agar dapat mengelompokkan unit-unit yang hampir sama dalam satu area tertentu. Pembelajaran ini biasanya sangat cocok untuk klasifikasi pola. Contoh algoritme jaringan saraf tiruan yang menggunakan metode *unsupervised* ini adalah c*ompetitive*, Kohonen, *Learning Vector Quantization* (LVQ).

2.8 *Backpropagation*

Backpropagation merupakan salah satu metode pembelajaran terawasi (*supervised learning*). Metode ini memiliki karakteristik meminimalkan nilai *error* pada *output* yang dihasilkan. Terdapat 3 tahapan dalam metode *backpropagation* yaitu *feedforward*, *backpropagation* dan penyesuaian bobot. Dalam metode *backpropagation*, terdapat beberapa persamaan yang digunakan untuk proses pembelajaran yang dilakukan (Fausset, 1994).

Persamaan 2.5 digunakan untuk menghitung nilai z_in_j. Nilai ini diperoleh dari hasil penjumlahan dari perkalian bobot v_{ij} dengan *input* (x_i).

$$z\_in_j = v_{oj} + \sum_i x_i v_{ij} \qquad (2.5)$$

Persamaan 2.6 adalah fungsi aktivasi untuk menghitung nilai z_j dengan masukan nilai z_in_j yang dihasilkan persamaan 2.4.

$$z_j = f(z\_in_j) \qquad (2.6)$$

Persamaan 2.7 digunakan untuk nilai y_in_k. Nilai ini diperoleh dari hasil penjumlahan dari perkalian fungsi aktivasi (z_j) dengan bobot (w_{ij}).

$$y\_in_k = w_{oj} + \sum_i z_j w_{ij} \quad (2.7)$$

Persamaan 2.8 merupakan fungsi aktivasi untuk menghitung nilai y_k, yang diperoleh dengan memasukkan nilai dari y_in_k pada Persamaan 2.7.

$$y_k = f(y\_in_k) \quad (2.8)$$

Aktivasi sigmoid biner

$$f_1(x) = \frac{1}{1+\exp^{(-x)}} \quad (2.9)$$

Aktivasi sigmoid bipolar

$$f_2(x) = \frac{1}{1+\exp^{(-x)}} - 1 \quad (2.10)$$

Persamaaan 2.11 digunakan untuk menghitung nilai δ_k pada *backpropagation*. Nilai ini menunjukkan *error* pada *node output*.

$$\delta_k = (t_k - y_k) f'(y\_in_k) \quad (2.11)$$

Persamaan 2.12 digunakan untuk menghitung kebenaran nilai bobot untuk kemudian diperbaiki (*update*) nilai dari w_{jk}.

$$\Delta w_{jk} = \alpha \delta_k z_j \quad (2.12)$$

Persamaan 2.13 digunakan untuk menghitung kebenaran nilai bias untuk kemudian diperbaiki (*update*) nilai dari w_{0k}.

$$\Delta w_{0k} = \alpha \delta_k \quad (2.13)$$

Persaman 2.14 digunakan untuk menghitung faktor δ di lapisan tersembunyi

$$\delta\_in_j = \sum_{k=1}^{m} \delta_k w_{jk} \quad (2.14)$$

Persamaan 2.15 merupakan fungsi aktivasi dengan masukan nilai δ_in_j pada Persamaan 2.14

$$\delta_j = \delta\_in\ f'(z\_in_j) \quad (2.15)$$

Persamaan 2.16 digunakan untuk menghitung perubahan bobot (v_{ij})

$$\Delta v_{ij} = \alpha \delta_j x_i \quad (2.16)$$

Persamaan 2.17 digunakan untuk menghitung perubahan bias (v_{0j})

$$\Delta v_{io} = \alpha \delta_j \quad (2.17)$$

Persamaan 2.18 digunakan untuk menghitung bobot (w_{jk}) baru dengan menjumlah w_{jk} lama dengan perubahan nilai Δw_{jk}

$$w_{jk}(baru) = w_{jk}(lama) + \Delta w_{jk} \quad (2.18)$$

Persamaan 2.19 digunakan untuk menghitung bobot (v_{ij}) baru dengan menjumlah v_{ij} lama dengan perubahan nilai Δv_{ij}

$$w_{0k}(baru) = w_{0k}(lama) + \Delta w_{0k} \quad (2.19)$$

Pelatihan dilakukan secara berulang sampai mencapai nilai *error* minimum atau iterasi maksimal tercapai.

Tahapan Proses *Backpropagation*

1. Inisialisasi bobot.
2. Jika kondisi berhenti belum terpenuhi, lakukan tahap 3-10
3. Untuk tiap pasangan *training*, lakukan tahap 4-9
4. Penghitungan *feedforward*

   Setiap *node* di lapisan *input* (X_i) menerima sinyal *input* dan mengirimkan sinyal tersebut ke semua *node* dalam lapisan tersembunyi
5. Setiap *node* di lapisan tersembunyi (Z_j) menggunakan Persamaan 2.5 dan mengunakan fungsi aktivasi pada Persamaan 2.6 untuk mengirim sinyal ke lapisan *output*
6. Setiap *node output* (Y_k) menjumlahkan bobot sinyal *input* menggunakan Persamaan 2.7 dan menggunakan Persamaan 2.8 untuk menghitung sinyal *output*
7. Setiap *node output* (Y_k) menghitung *error* menggunakan Persamaan 2.10, menghitung kesesuaian bobot dengan Persamaan 2.12 dan menghitung kesesuaian bias dengan Persamaan 2.13 dan mengirim δ_k *node* dalam *layer*nya
8. Setiap *node* di lapisan tersembunyi (Z_j) menjumlah delta *input* menggunakan Persamaan 2.14, menghitung *error* dengan Persamaan 2.15, menghitung kesesuaian bobot dengan Persamaan 2.16 dan bias dengan Persamaan 2.17.
9. Update *node output* memperbarui bobot dan bias dengan Persamaan 2.18 dan tiap unit di lapisan tersembunyi memperbarui bobot dan bias melalui Persamaan 2.19
10. Menguji kondisi berhenti

2.9 Normalisasi Data

Untuk mempermudah melakukan peritungan agar data selalu dalam rentang yang tidak terlalu besar maka dilakukan proses normalisasi data. Metode yang digunakan pada penelitian ini adalah metode Min-Max. Persamaan yang digunakan pada proses normalisasi data yang dilakukan adalah Persamaan 2.20.

$$x' = \left(0{,}8 * \frac{x - \min}{\max - \min} \right) + 0{,}1 \quad (2.20)$$

Keterangan:

X : data asli

X’ : data hasil normalisasi

Min : nilai minimal data

Max : nilai minimal data

2.10 Denormalisasi Data

Proses denormalisasi data merupakan proses kebalikan dari normalisasi data. Hasil dari proses denormalisasi data merupakan nilai data asli bukan berada dalam rentang tertentu. Persamaan yang digunakan adalah Persamaan 2.21.

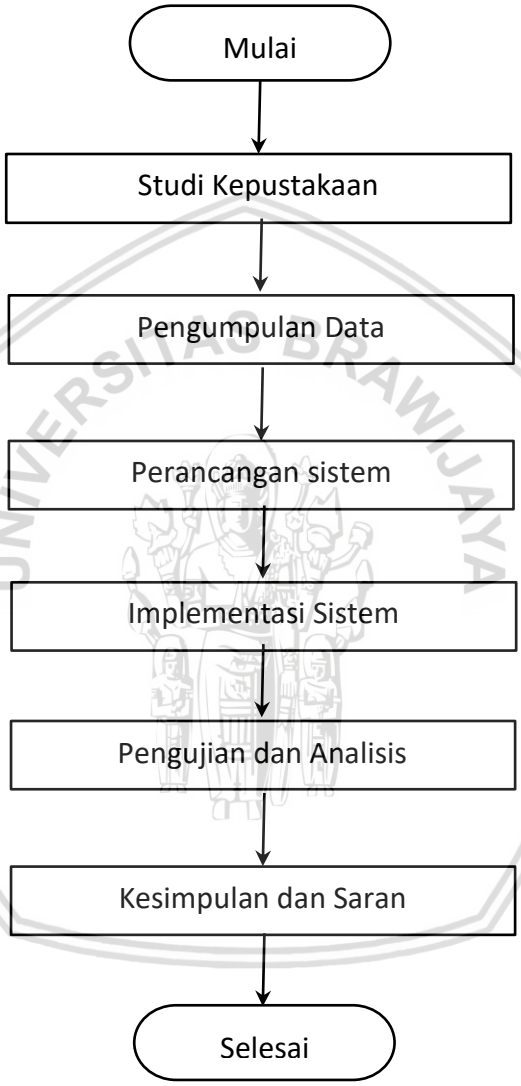
$$x'' = \left(\frac{(\max - \min) * (x' - 0{,}1)}{0{,}8} \right) + \min \quad (2.21)$$

Keterangan:

X” : data hasil denormalisasi

BAB 3 METODOLOGI

Bab ini menjelaskan tentang tahapan-tahapan yang dilakukan dalam penelitian berkaitan dengan peramalan debit bendungan menggunakan metode *backpropagation* dan algoritme genetika. Tahapan yang dilakukan pada penelitian ini dapat diilustrasikan seperti Gambar 3.1.



Gambar 3.1 Tahapan Penelitian

3.1 Studi Kepustakaan

Tahap awal yang dilakukan pada penelitian ini adalah melakukan studi kepustakaan. Tahap ini melakukan pengumpulan penelitian yang pernah dilakukan dan mempelajari dasar teori yang terkait untuk digunakan sebagai acuan dasar dalam penelitian. Teori yang berkaitan dengan penelitian ini, yaitu:

a. Algoritme Genetika
b. Jaringan Saraf Tiruan
c. Metode *Backpropagation*

Literatur yang menjadi acuan pada penelitian ini didapatkan dari jurnal, skripsi, dan buku yang berkaitan dengan penelitian.

3.2 Analisis Kebutuhan

Untuk mendefinisikan kebutuhan apa saja yang akan digunakan pada penilitian peramalan debit bendungan menggunakan metode *backpropagation* dan algoritme genetika. Dilakukan proses analisis kebutuhan. Kebutuhan-kebutuhan ini meliputi

1. Sistem mampu menampilkan hasil peramalan debit bendungan berdasarkan data *time* series yang ada.
2. Sistem mampu menampilkan tingkat akurasi dari hasil peramalan terhadap data aktual.

3.3 Pengumpulan data

Tahap selanjutnya dari penelitian ini adalah pengumpulan data. Data yang dikumpulkan adalah data hidrologi mengenai debit inflow Bendungan Sutami. Data didapatkan dari Perum Jasa Tirta I selaku pengelola Bendungan Sutami. Data yang berhasil dikumpulkan adalah data debit rata-rata per bulan. Data yang digunakan pada penelitian ini adalah data debit bendungan mulai tahun 2008 sampai dengan tahun 2017. Data-data yang ada tersebut nantinya akan dilakukan proses normalisasi untuk kemudian dilakukan perhitungan dalam algoritme genetika dan jaringan saraf tiruan. Setelah diperoleh hasil peramalan, maka dapat akan didenormalisasi sehingga data peramalan debit bendungan bulan berikutnya dapat diketahui.

3.4 Perancangan sistem

Dalam tahap perancangan sistem akan dijelaskan mengenai gambaran umum sistem yang akan dibangun, diagram alir sistem, perhitungan manual, perancangan antarmuka dan perancangan pengujian.

3.5 Implementasi

Pada tahap ini dilakukan implementasi sesuai dengan perancangan yang telah dibuat. Implementasi yang dilakukan adalah implementasi perangkat keras, implementasi perangkat lunak, implementasi antarmuka, dan implementasi algoritme genetika dan metode *backpropagation* untuk peramalan debit bendungan.

3.6 Pengujian dan Analisis

Pada tahap ini, pada aplikasi yang telah dibuat akan dilakukan pengujian sesuai skenario yang telah ditentukan. Tahap berikutnya adalah melakukan analisis berdasarkan data hasil pengujian.

3.7 Kesimpulan dan Saran

Tahap terakhir dari penelitian ini adalah mengambil kesimpulan dan memberikan saran. Pengambilan kesimpulan berdasarkan hasil pengujian dan analisis sistem yang dibangun untuk melakukan peramalan debit bendungan menggunakan metode *backpropagation* dan algoritme genetika. Saran yang diberikan diharapkan akan menambah nilai tambah untuk penelitian lanjutan yang akan dilakukan baik menggunakan metode ataupun kasus yang terkait.

BAB 4 PERANCANGAN

4.1 Formulasi Permasalahan

Penelitian ini mengambil permasalahan mengenai peramalan debit bendungan. Kebutuhan akan adanya data untuk merencanakan pola operasi pintu air di bendungan merupakan salah satu sebab perlunya peramalan debit bendungan dilakukan. Dari data prediksi yang dihasilkan dapat direncanakan pula listrik yang dihasilkan melalui PLTA dan rencana irigasi ke area persawahan.

4.2 Deskripsi Data

Data yang digunkan pada penelitian ini adalah data debit bulanan Bendungan Sutami. Historis data dimulai dari tahun 2008 sampai tahun 2017. Satuan dari debit adalah m3/detik yang menunjukkan berapa jumlah air yang masuk pada bendungan setiap detiknya. Contoh data debit yang didapatkan seperti pada Tabel 4.1.

Tabel 4.1 Contoh data debit bendungan

Bulan	Debit
Januari-2008	128,51
Februari-2008	122,31
Maret-2008	105,00
April-2008	90,13
Mei-2008	63,68

4.3 Perancangan Sistem

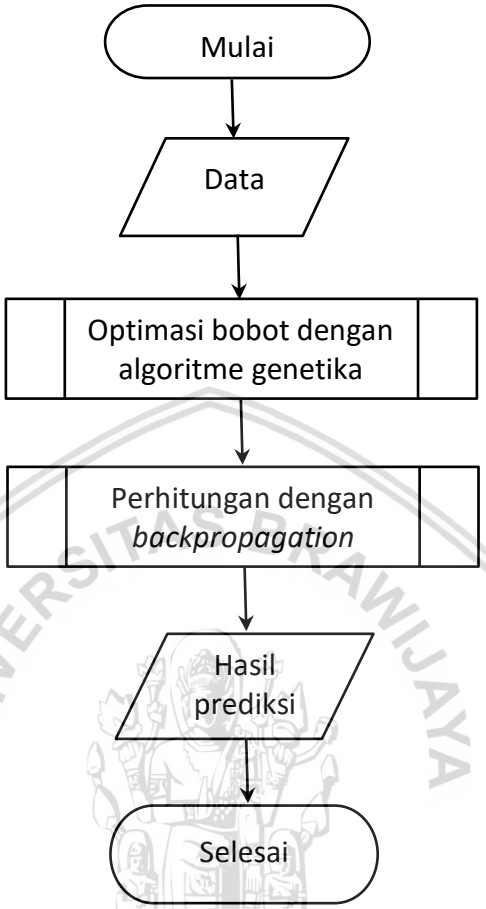
4.3.1 Gambaran Umum

Sistem yang akan dibangun pada penelitian ini merupakan aplikasi untuk melakukan peramalan debit bendungan. Data debit rata-rata akan dijadikan nilai masukan. Proses perhitungan yang dilakukan aplikasi nantinya adalah optimasi bobot awal yang dilakukan menggunakan algoritme genetika dan bobot hasil dari algoritme genetika tersebut akan dijadikan nilai dalam inisialisasi bobot awal dalam proses pembelajaran menggunakan metode *backpropagation*. Hasil dari aplikasi adalah peramalan debit untuk bulan yang akan datang.

4.3.2 Diagram Alir Sistem

Tahapan sistem untuk melakukan peramalan debit bendungan menggunakan metode *backpropagation* dan algortima genetika secara umum diilustrasikan dalam Gambar 4.1. Tahapan dilakukan yaitu, input data debit bendungan, optimasi bobot awal menggunakan algoritme genetika, proses perhitungan

menggunakan metode *backpropagation*, menampilkan keluaran sistem berupa data hasil prediksi debit bendungan.

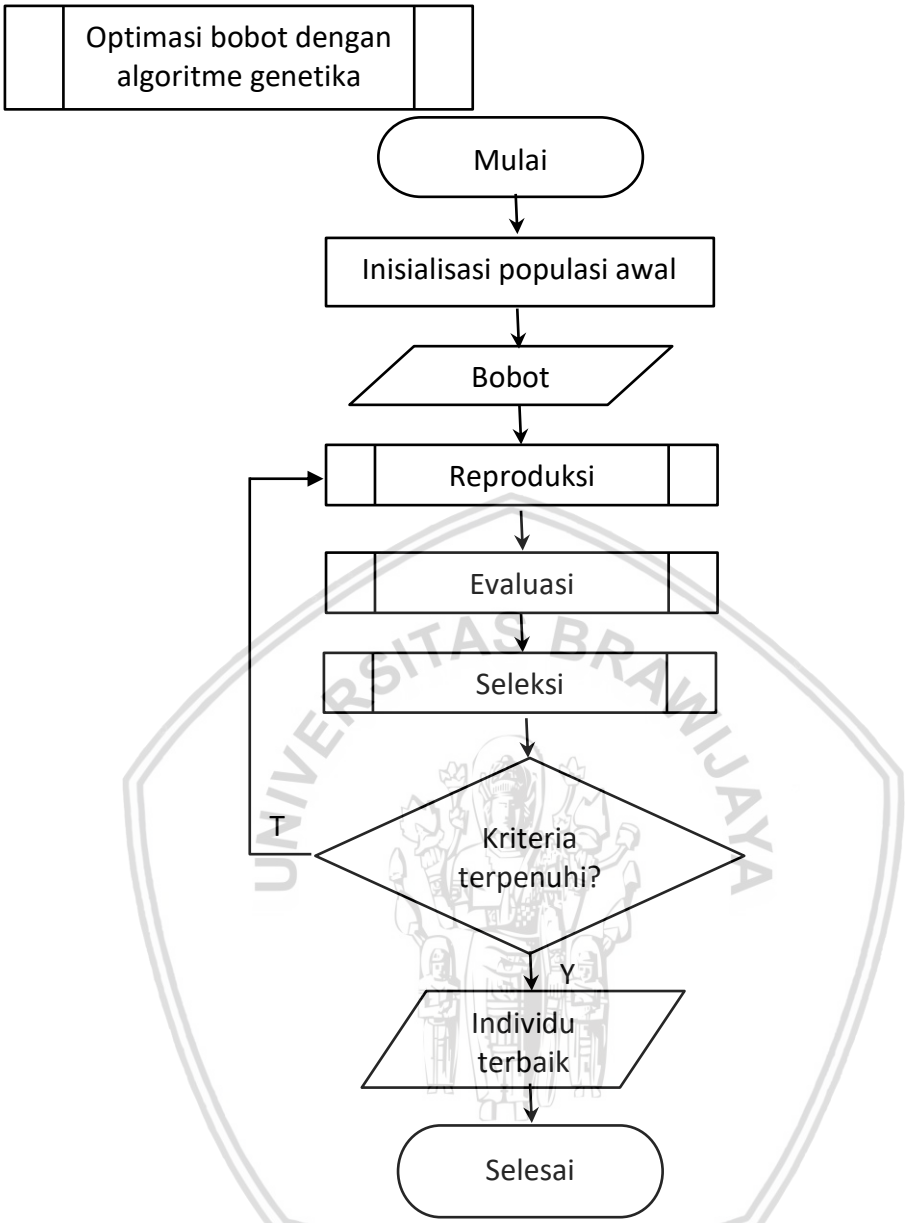


Gambar 4.1 Diagram Alir Sistem

Penjelasan mengenai tahapan yang dilakukan dalam peramalan debit bendungan menggunakan metode *backpropagation* dan algoritme genetika adalah sebagai berikut:

1. Data masukan adalah data debit bulanan yang akan digunakan dalam proses peramalan.
2. Melakukan optimasi bobot dan bias menggunakan algoritme genetika
3. Setelah bobot dan bias didapatkan, nilai tersebut menjadi parameter yang digunakan dalam inisialisasi bobot awal dan bias dalam proses perhitungan menggunakan metode *backpropagation*.
4. Sistem akan menghasilkan hasil keluaran berupa data hasil peramalan debit bendungan

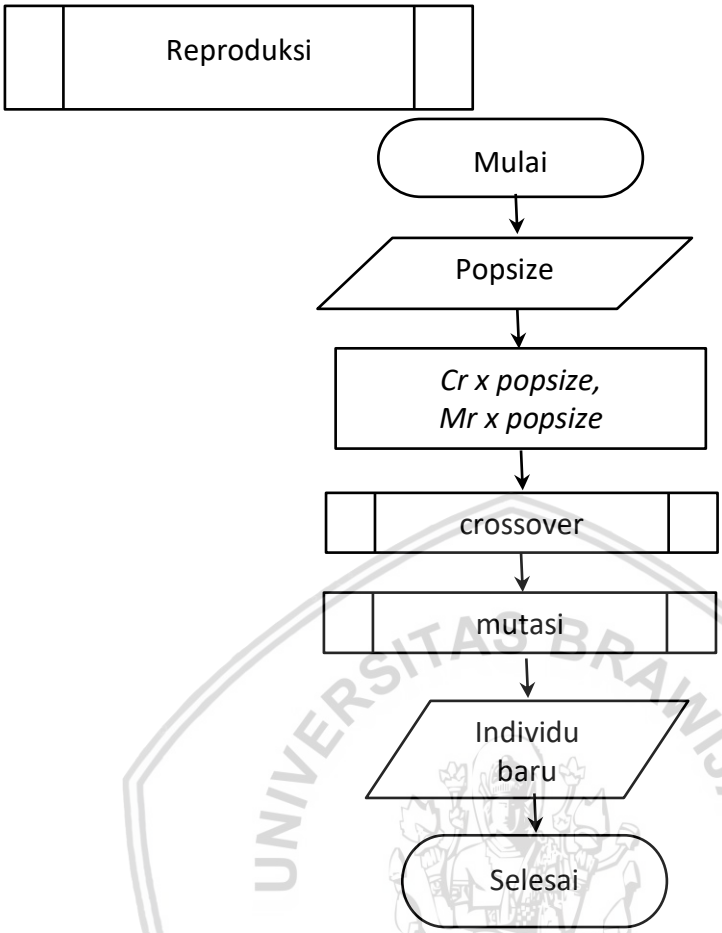
Untuk optimasi bobot dan bias menggunakan algoritme genetika, diagram alirnya adalah seperti pada Gambar 4.2.



Gambar 4.2 Diagram Alir Algoritma Genetika untuk Optimasi Bobot

Penjelasan proses algoritme genetika dalam melakukan optimasi bobot dan bias dalam Gambar 4.2 adalah sebagai berikut:

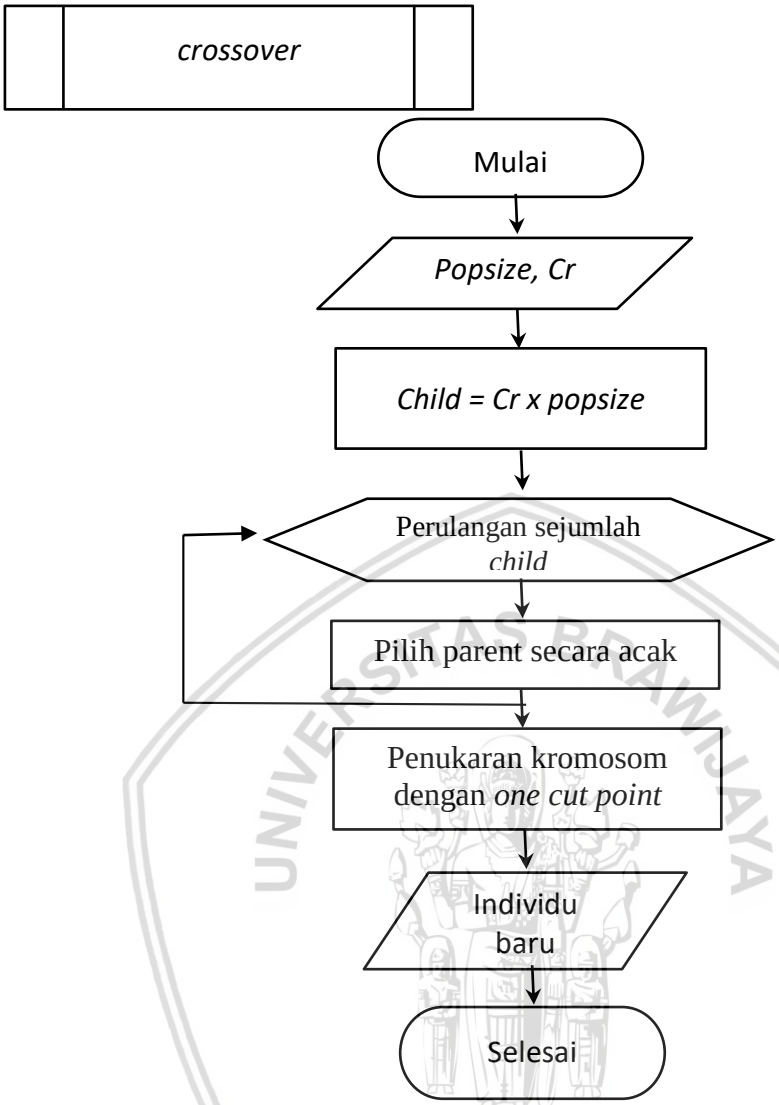
1. Dilakukan inisialisasi populasi awal secara acak.
2. Nilai bobot awal dan bias adalah nilai masukan algoritme genetika.
3. Dilakukan proses reproduksi (*crossover* dan mutasi).
4. Dilakukan evaluasi untuk mendapatkan nilai *fitness*.
5. Melakukan seleksi individu
6. Jika sudah memenuhi kriteria, maka sistem akan menghasilkan individu terbaik terkait bobot dan bias, namun bila belum sesuai kriteria, tahapan akan kembali ke proses reproduksi.



Gambar 4.3 Diagram alir proses reproduksi

Penjelasan proses reproduksi dalam algoritme genetika dalam Gambar 4.3 adalah sebagai berikut:

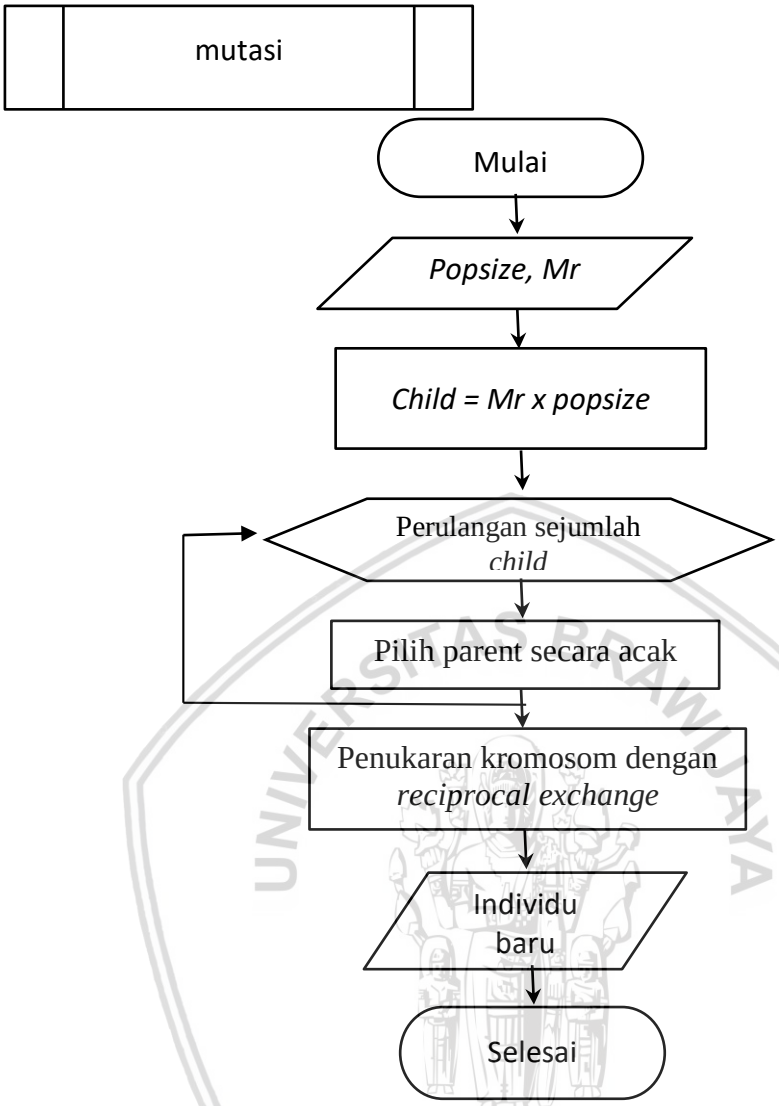
1. Proses mendapat masukan ukuran popsize.
2. Ukuran popsize dikalikan dengan variabel Cr dan Mr untuk menghitung jumlah keturunan yang akan dihasilkan.
3. Dilakukan proses *crossover* dan mutase.
4. Dihasilkan individu hasil proses reproduksi.
5. Melakukan seleksi individu



Gambar 4.4 Diagram alir proses *crossover*

Penjelasan proses *crossover* dalam Gambar 4.4 adalah sebagai berikut:

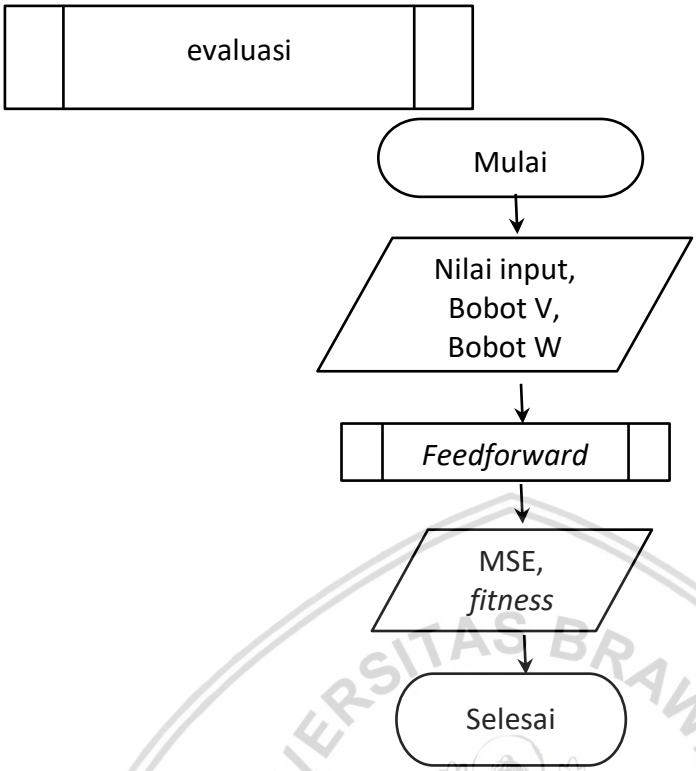
1. Proses mendapat masukan ukuran *popsize* dan cross*over rate* (Cr).
2. Ukuran *popsize* dikalikan dengan variabel Cr untuk menghitung jumlah keturunan yang akan dihasilkan.
3. Dilakukan perulangan sejumlah hasil keturunan, dengan memilih *parent* secara acak.
4. Melakukan penukaran kromosom induk dengan metode *one cut point*.
5. Dihasilkan individu baru hasil *crossover*.



Gambar 4.5 Diagram alir proses mutasi

Penjelasan proses mutasi dalam Gambar 4.5 adalah sebagai berikut:

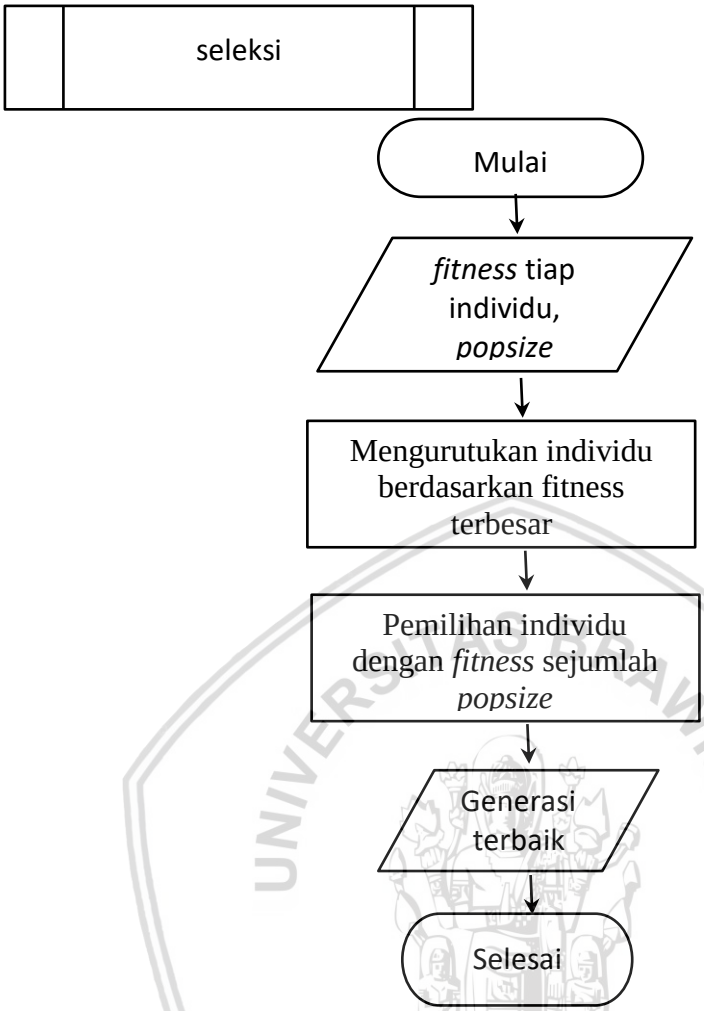
1. Proses mendapat masukan ukuran *popsize* dan *mutation rate* (Mr).
2. Ukuran *popsize* dikalikan dengan variabel Mr untuk menghitung jumlah keturunan yang akan dihasilkan.
3. Dilakukan perulangan sejumlah hasil keturunan, dengan memilih *parent* secara acak.
4. Melakukan penukaran kromosom induk dengan metode *reciprocal exchange*.
5. Dihasilkan individu baru hasil mutasi.



Gambar 4.6 Diagram alir proses evaluasi

Penjelasan proses evaluasi dalam Gambar 4.6 adalah sebagai berikut:

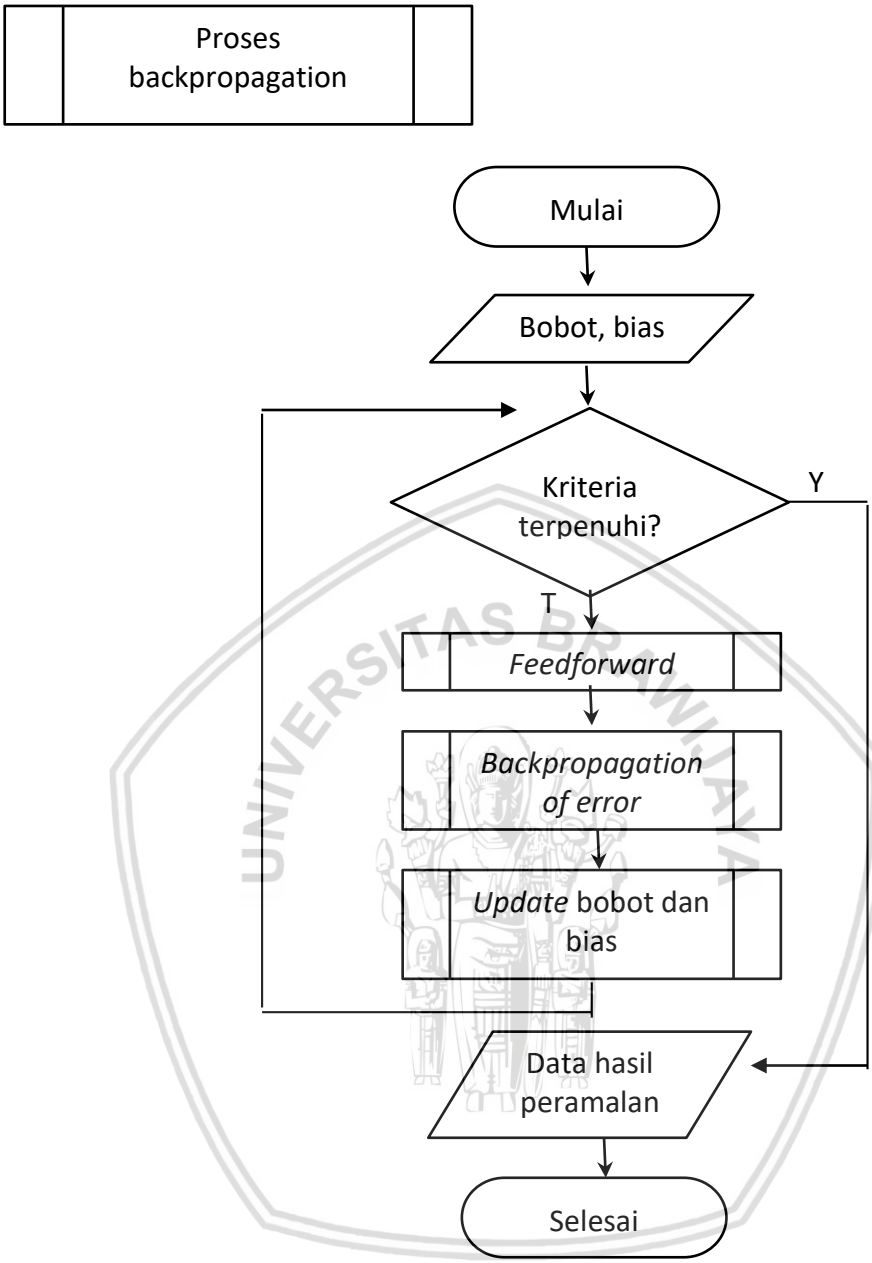
1. Proses mendapat masukan nilai input, bobot V dan bobot W.
2. Dilakukan proses *feedforward*.
3. Dihasilkan nilai MSE dan nilai fitness untuk masing-masing individu



Gambar 4.7 Diagram alir proses seleksi

Penjelasan proses seleksi dalam Gambar 4.7 adalah sebagai berikut:

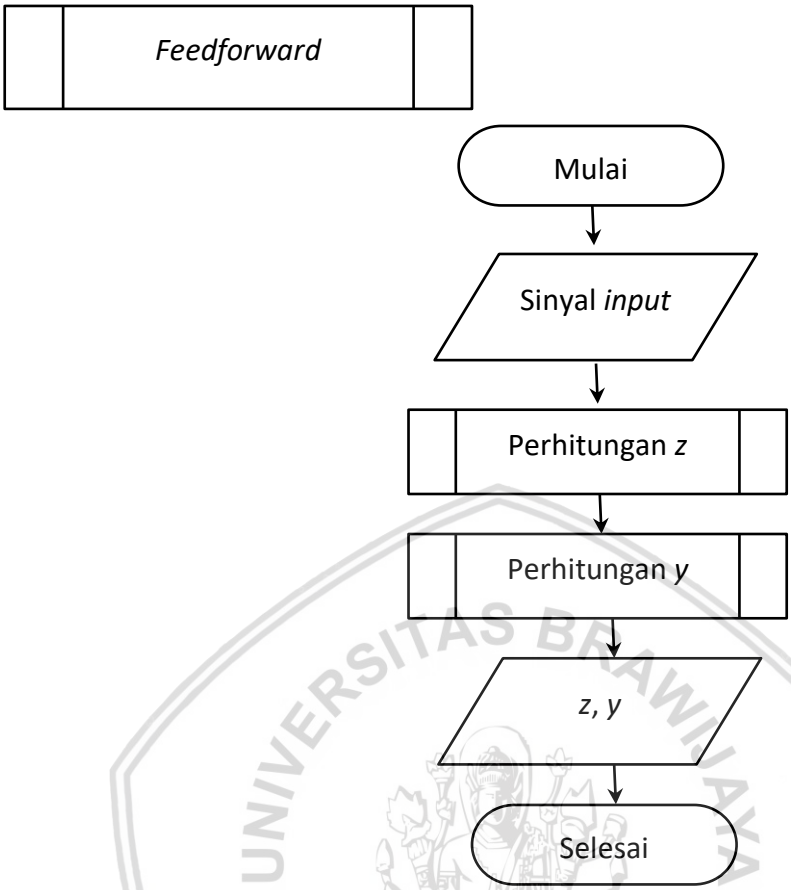
1. Proses mendapat masukan berupa *fitness* dari masing-masing individu hasil proses evaluasi dan ukuran *popsize.*
2. Masing-masing individu tersebut kemudian diurutkan berdasarkan nilai fitness terbesar.
3. Dari pengurutan tersebut kemudian diambil individu teratas sejumlah ukuran popsize.
4. Didapatkan hasil generasi terbaik untuk dilanjutnkan ke tahap berikutnya.



Gambar 4.8 Diagram alir proses *backpropagation*

Penjelasan proses *backpropagation* dalam Gambar 4.8 adalah sebagai berikut:

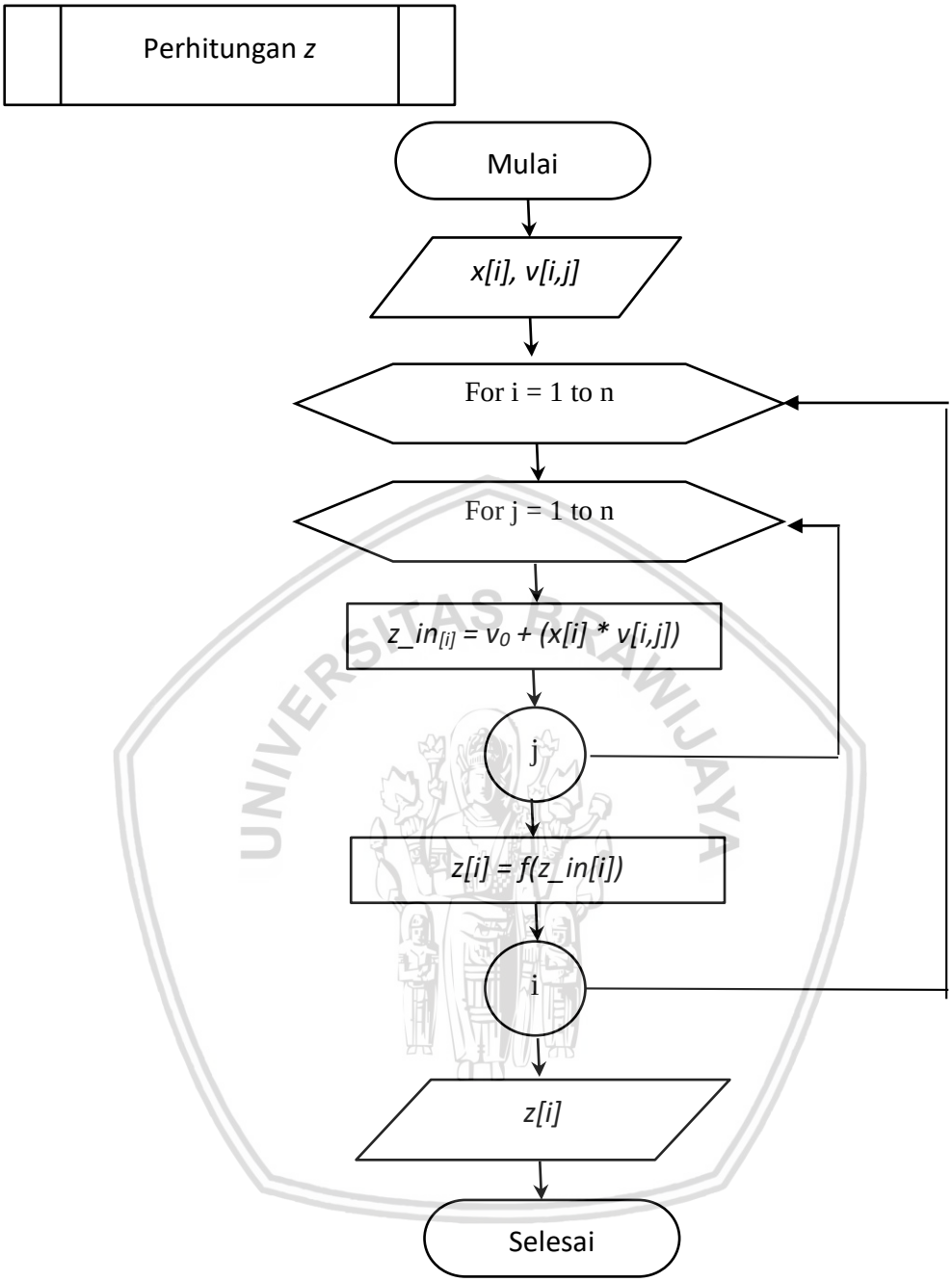
1. Sistem mendapat masukan berupa bobot dan bias hasil dari algoritme genetika.
2. Pengecekan terhadap kriteria yang ditentukan, jika sudah memenuhi maka sistem menampilkan data hasil peramalan, jika belum maka akan dilanjutkan pada proses *feedforward*.
3. Sistem melakukan pehitungan *backpropagation error*.
4. Sistem melakukan *update* bobot dan bias.



Gambar 4.9 Diagram alir proses *feedforward*

Gambar 4.9 menunjukkan tahapan yang dilakukan dalam proses feedforward. Penjelasan dari tahapan tersebut adalah sebagai berikut:

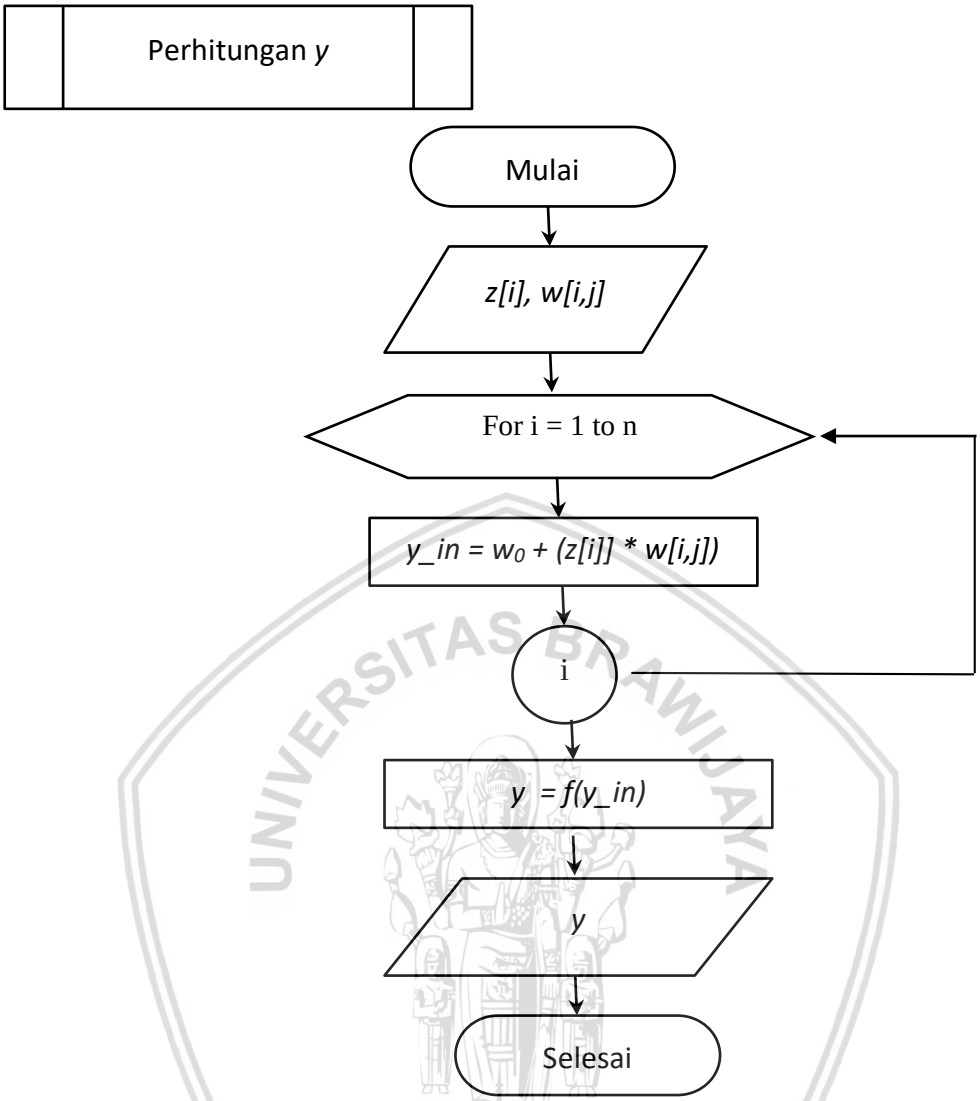
1. Sistem menerima sinyal *input x*. Nilai *x* merepresentasikan nilai debit pada bulan tertentu.
2. Sistem melakukan perhitungan untuk mencari nilai *output* dan *hidden layer*.
3. Sistem melakukan perhitungan untuk mencari nilai *output* (*y*).
4. Sistem menghasilkan keluaran berupa nilai *z* dan *y.*



Gambar 4.10 Diagram alir perhitungan nilai Z_i

Gambar 4.10 menunjukkan tahapan yang dilakukan untuk mencari nilai *Z*. Penjelasan dari tahapan tersebut adalah sebagai berikut:

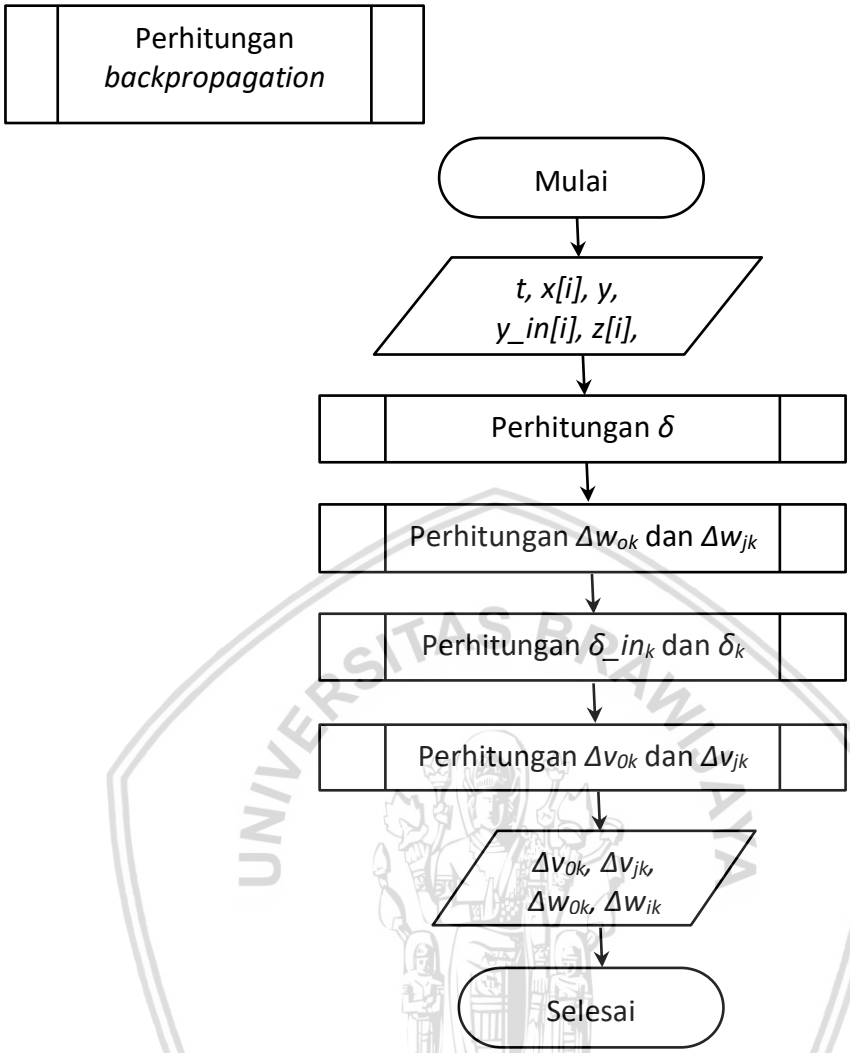
1. Sistem menerima masukan sinyal *input* dari unit (*x*) dan bobot (*v*)
2. Sistem melakukan perulangan untuk menghitung nilai sinyal *input* yang sudah memilki bobot (z_*in*) dan menghitung nilai sinyal *input* dari unit *input* menggunakan fungsi aktivasi z = *f(z_in)*
3. Sistem menghasilkan *output* nilai z.



Gambar 4.11 Diagram alir perhitungan nilai y

Gambar 4.11 menunjukkan tahapan yang dilakukan untuk mencari nilai *output* (*y*). Berikut Penjelasan dari tahapan tersebut adalah sebagai beikut:

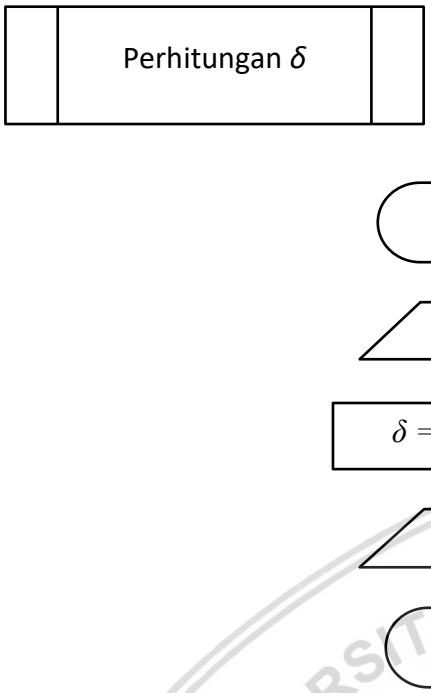
1. Sistem menerima *input* berupa sinyal *output* dari *neuron* pada *hidden layer* (*z*) dan bobor (*w*)
2. Sistem melakukan perulangan untuk menghitung nilai *input* yang sudah memiliki bobot (*y_in*) dan menghitung nilai *output* dari *neuron* pada *output* menggunakan fungsi aktivasi y = *f(y_in).*
3. Sistem menghasilkan *output* nilai *y*.



Gambar 4.12 Diagram alir perhitungan *backpropagation*

Gambar 4.12 menunjukkan tahapan dalam perhitungan *backpropagation*. Penjelasan dari tahapan tersebut adalah sebagai berikut:

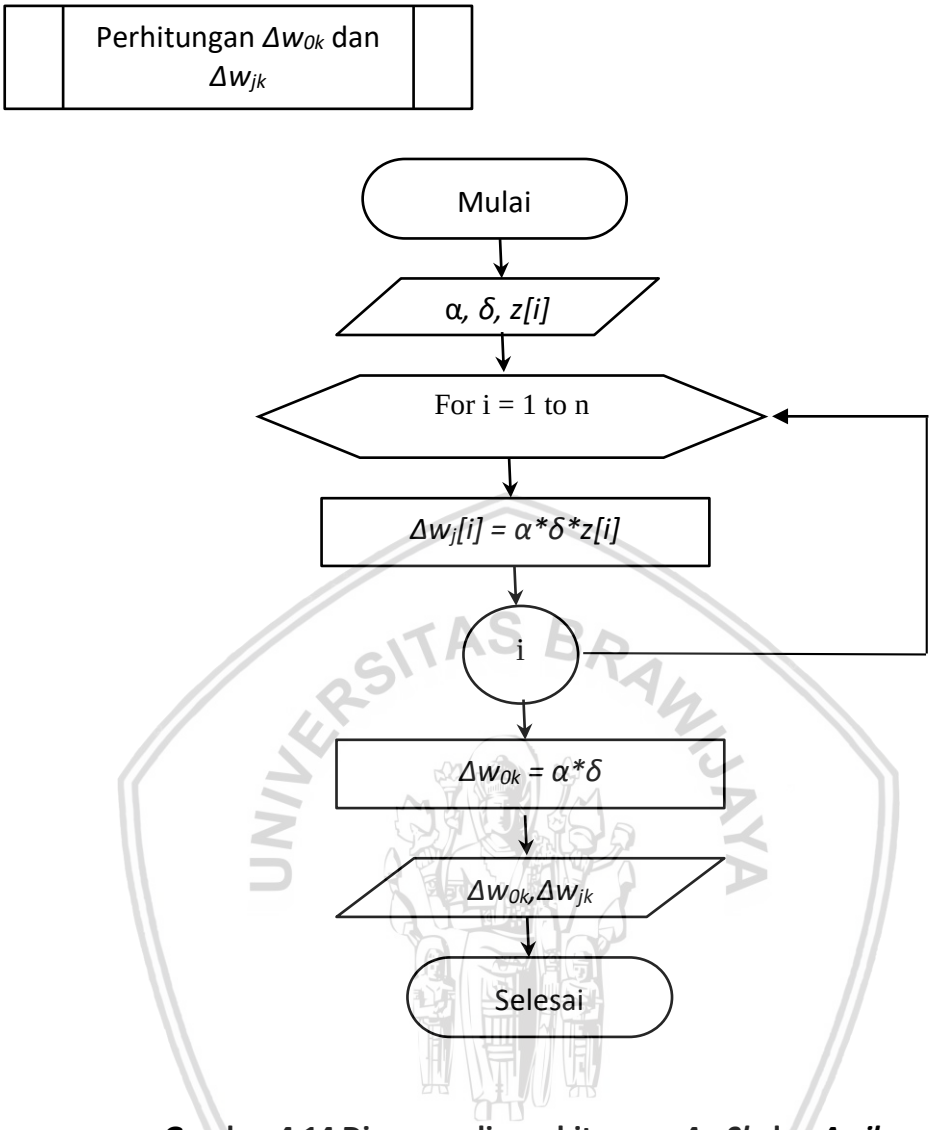
1. Sistem menerima *input* berupa *x[i], y, y_in[i], z[i], z_in[i].*
2. Sistem melakukan perhitungan nilai δ dengan tujuan mencari tingkat kesalahan antara *output* dan target.
3. Sistem melakukan perhitungan Δw_{ok} dan Δw_{jk} yang betujuan untuk mencari tingkat kesalahan bobot dan bias. Nilai ini akan digunakan untuk memperbarui bobot *w* pada tahap *update* bobot dan bias.
4. Sistem melakukan perhitungan nilai δ_in_k dan δ_k.
5. Sistem melakukan perhitungan Δv_{0k} dan Δv_{jk} yang betujuan untuk mencari tingkat kesalahan bobot dan bias. Nilai ini akan digunakan untuk memperbarui bobot *v* pada tahap *update* bobot dan bias.
6. Sistem menghasilkan *output* berupa nilai Δv_{0k}, Δv_{jk}, Δw_{0k}, Δw_{jk}.



Gambar 4.13 Diagram alir perhitungan *δ*

Gambar 4.13 menunjukkan tahapan yang dilakukan dalam pehitungan nilai *δ*. Penjelasan dari tahapan tersebut adalah sebagai berikut:

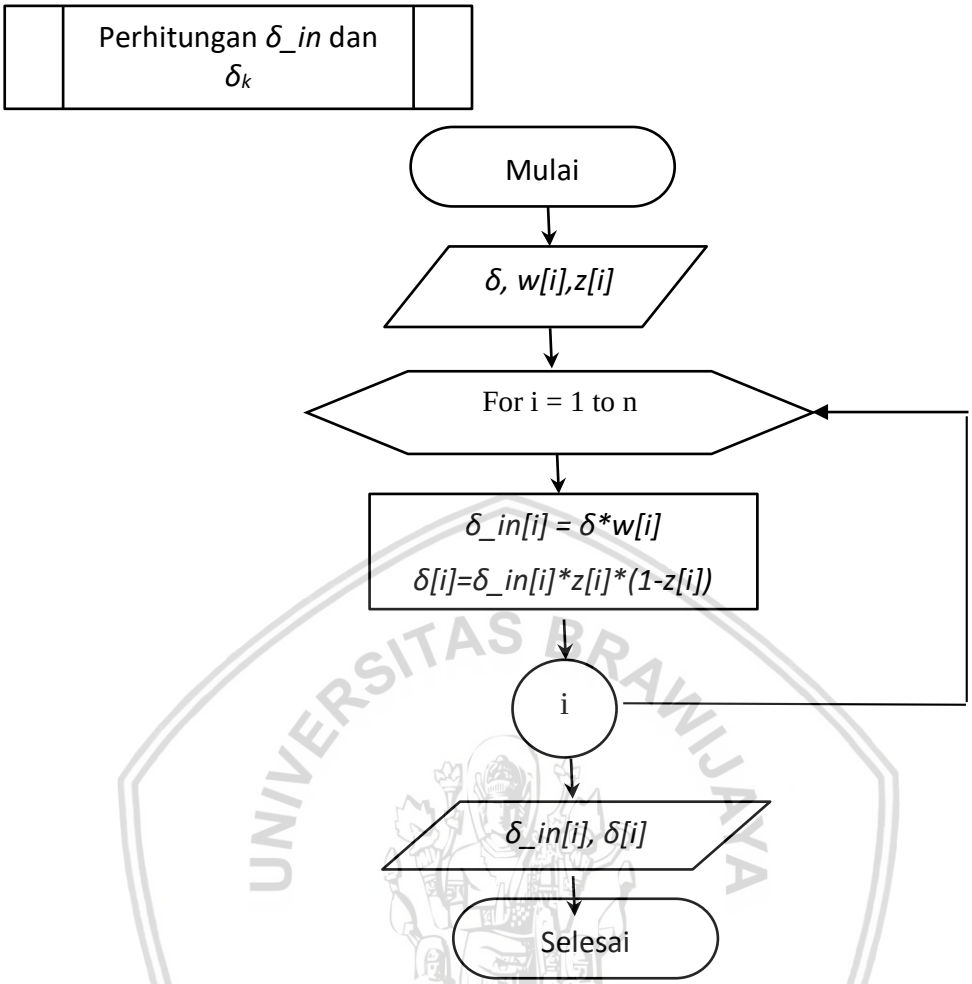
1. Sistem menerima *input* nilai *t* dan *y*.
2. Sistem melakukan perhitungan nilai *δ*.
3. Sistem menghasilkan *output* berupa nilai *δ*.



Gambar 4.14 Diagram alir perhitungan *Δw0k* dan *Δwjk*

Gambar 4.14 menunjukkan tahapan dalam proses perhitungan Δw_{0k} dan Δw_{jk}. Penjelasan Dari tahapan tersebut adalah sebagai berikut:

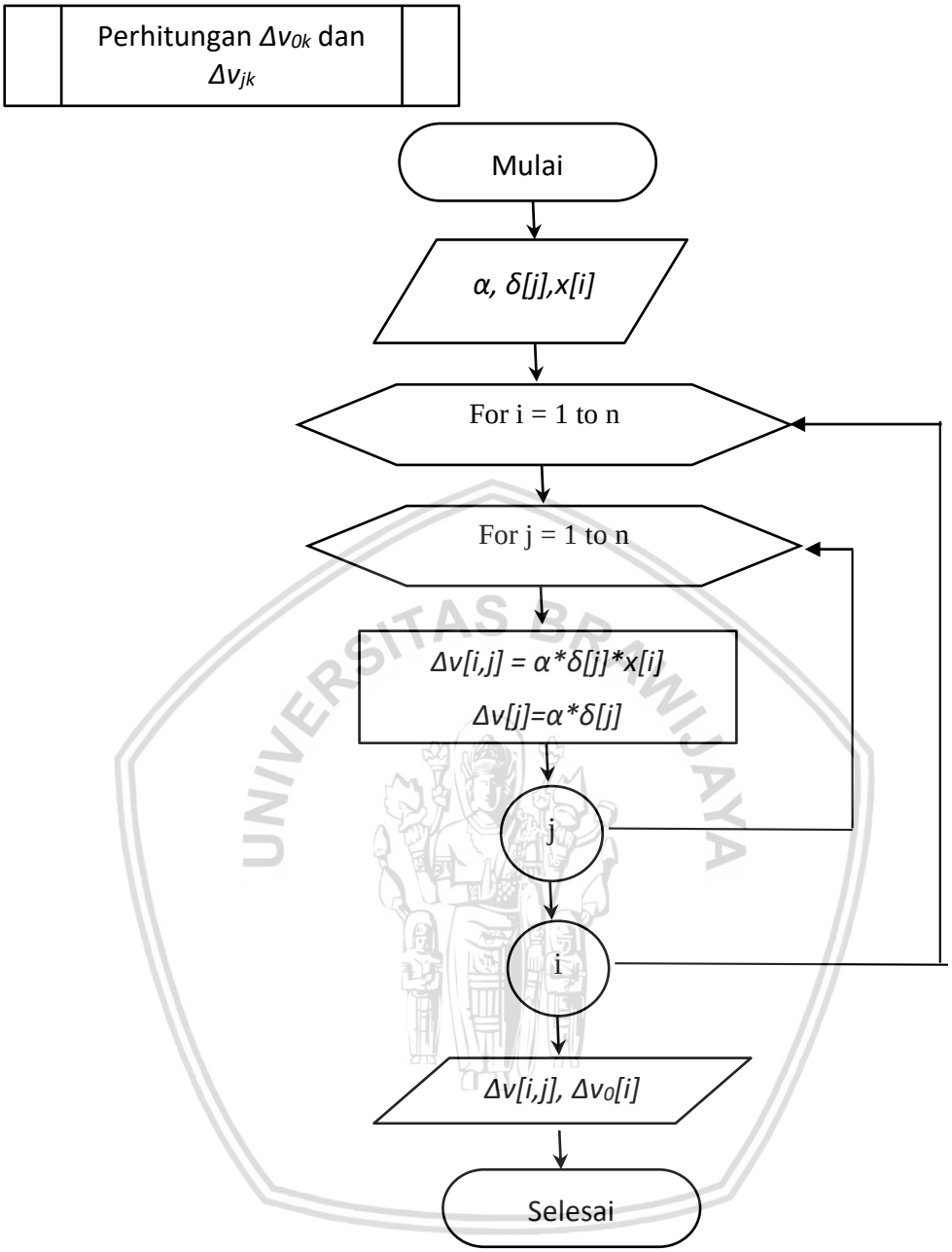
1. Sistem menerima *input* nilai *α, δ, z[i].*
2. Sistem melakukan perulangan untuk menghitung nilai Δw_{jk}.
3. Setelah perulangan selesai, dilakukan perhitungan nilai Δw_{0k}.
4. Sistem menghasilkan *output* berupa Δw_{0k} dan Δw_{jk}.



Gambar 4.15 Diagram alir perhitungan δ_*in* dan *δk*

Gambar 4.15 menunjukkan tahapan perhitungan *δ_in* dan δ_k. Penjelasan dari tahapan tersebut adalah sebagai berikut:

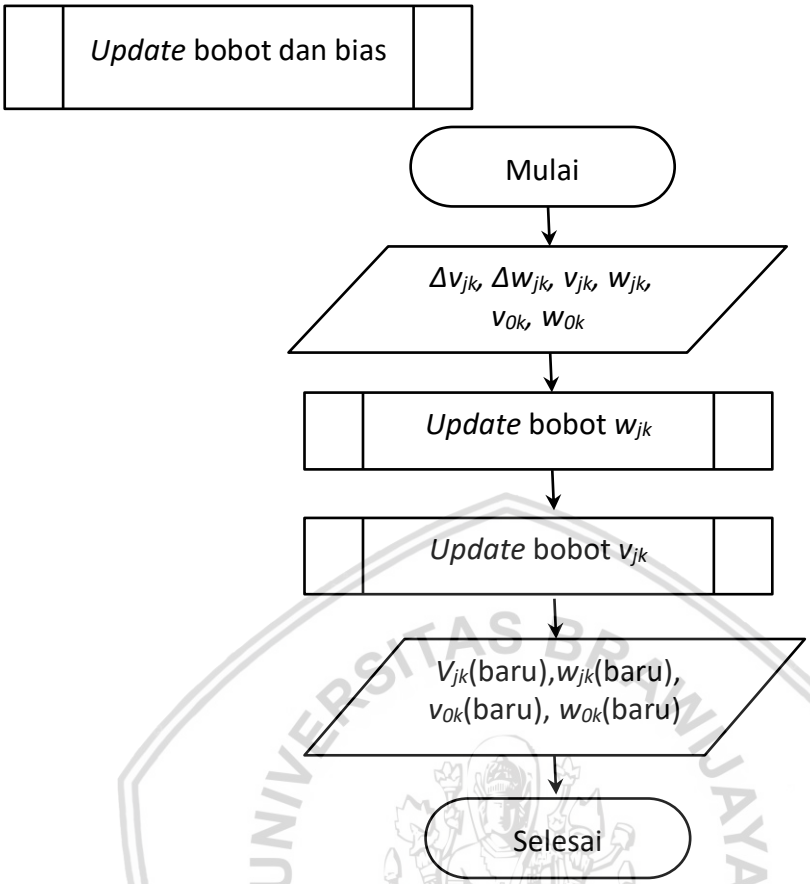
1. Sistem menerima *input δ, w[i],z[i].*
2. Sistem melakukan perhitungan δ_in_k dan δ_k.
3. Sistem menghasilkan *output* nilai *δ_in* dan δ_k.



Gambar 4.16 Diagram alir perhitungan *Δv0k* dan *Δvjk*

Gambar 4.16 menunjukkan tahapan perhitungan *Δv0k* dan *Δvjk*. Penjelasan tahapan tersebut adalah sebagai berikut:

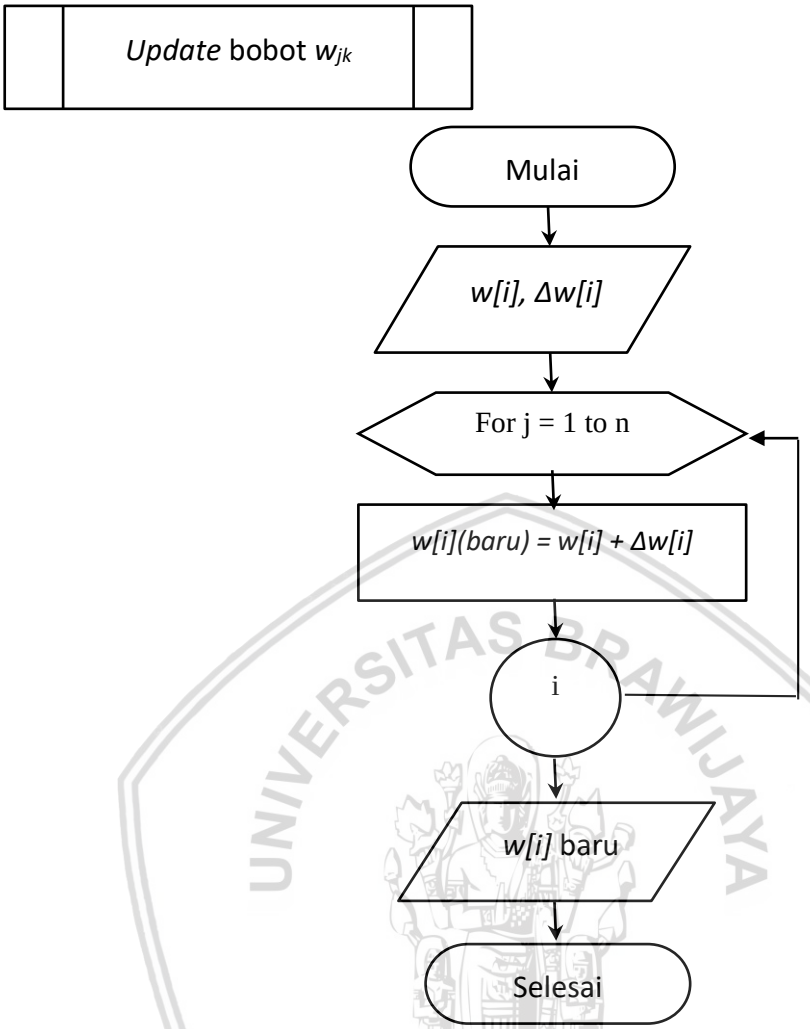
1. Sistem menerima *input α, δ[j],x[i].*
2. Sistem melakukan perulangan untuk menghitung Δv_{0k} dan Δv_{jk}.
3. Sistem menghasilkan *output* Δv_{0k} dan Δv_{jk}.



Gambar 4.17 Diagram alir *update* bobot dan bias

Gambar 4.17 menunjukkan tahapan dalam proses *update* bobot dan bias. Penjelasan tahapan tersebut adalah sebagai berikut:

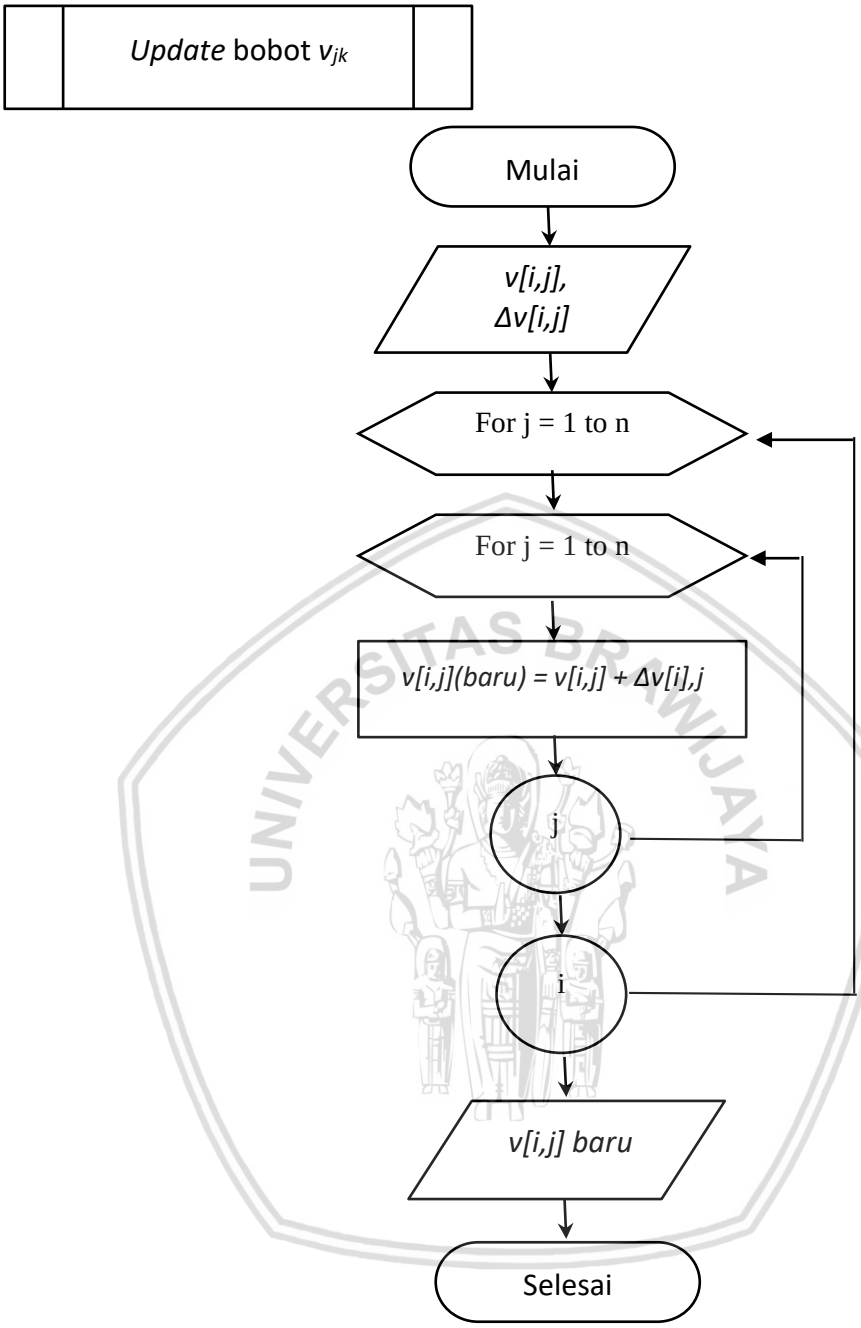
1. Sistem menerima *input*
2. Sistem melakukan perhitungan untuk *update* bobot $w_{jk.}$
3. Sistem melakukan perhitungan untuk *update* bobot $v_{jk.}$
4. Sistem menghasilkan *output* nilai bobot (v_{jk},w_{jk}) dan bias (w_{0k}, v_{0k}) yang baru.



Gambar 4.18 Diagral alir *update* bobot *wjk*

Gambar 4.18 menunjukkan tahapan dalam proses update bobot dan bias (w_{jk}, w_{0k}). Penjelasan tahapan tersebut adalah sebagai berikut:

1. Sistem menerima *input w[i], Δw[i]*.
2. Sistem melakukan perhitungan untuk *update* bobot (w_{jk}) dan bias (w_{0k}).
3. Sistem menghasilkan *output* nilai bobot (w_{jk}) dan bias (w_{0k}) yang baru.



Gambar 4.19 Diagram alir *update* bobot *vjk*

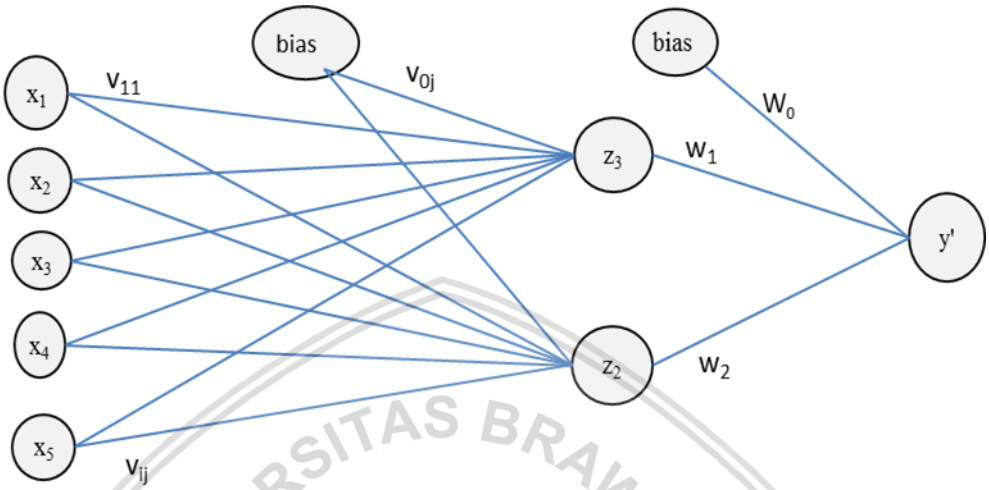
Gambar 4.19 menunjukkan tahapan dalam proses *update* bobot dan bias (v_{jk}, v_{0k}). Penjelasan tahapan tersebut adalah sebagai berikut:

1. Sistem menerima *input* v[i,j], Δv[i,j].
2. Sistem melakukan perhitungan untuk *update* bobot (v_{jk}) dan bias (v_{0k}).
3. Sistem menghasilkan *output* nilai bobot (v_{jk}) dan bias (v_{0k}) yang baru.

4.4 Perhitungan Manual

4.4.1 Representasi Kromosom

Pada perhitungan ini digunakan arsitektur jaringan saraf tiruan seperti pada Gambar 4.8



Gambar 4.20 Arsitektur jaringan saraf tiruan

Arsitektur jaringan saraf tiruan akan menentukan representasi kromosom dari tiap individu. Jumlah neuron pada input *layer* ditentukan oleh jumlah bulan yang akan digunakan sebagai input pada proses pelatihan. Jumlah neuron pada *hidden layer* akan menentukan jumlah bobot dan bias yang masuk ke *hidden layer*, selain itu akan menentukan juga jumlah bobot dan bias yang keluar dari *hidden layer* ke output *layer*. Arsitekur jaringan saraf tiruan yang akan digunakan pada perhitungan manual ini adalah 5 neuron input, 2 neuron *hidden* dan 1 output seperti yang telah digunakan pada penelitian mengenai peramalan debit sebelumnya (Assefa, 2016). Berdasarkan arsitektur tersebut, akan didapatkan reprentasi kromosom dengan jumlah gen sebanyak 25 gen. Representasi kromosom didapatkan dari seluruh nilai bobot pada arsitektur jaringan saraf tiruan pada Gambar 4.15 (v11, v12,.., v52) dan nilai biasnya (w0, w1, w2, w3, w4). Inisialisasi populasi (popSize) tediri dari 10 individu Untuk representasi kromosom individu dilakukan secara acak dengan rentang antara -1 sampai dengan 1 dengan representasi kromosom sebagai berikut yang mana bobot direpresentasikan dengan huruf v dan bias dengan huruf w:

Individu 1 (P1):

Tabel 4.2 Representasi kromosom individu 1

V01	V02	V11	V12	V21	V22	V31	V32	V41	V42	V51	V52
0,328	0,427	0,996	-0,988	0,372	-0,448	0,923	-0,839	-0,208	-0,026	-0,480	0,843
W0	W1	W2									
0,081	-0,101	0,011									

Individu 2 (P2):

Tabel 4.3 Representasi kromosom individu 2

V01	V02	V11	V12	V21	V22	V31	V32	V41	V42	V51	V52
-0,269	-0,371	0,482	-0,578	0,242	0,779	-0,853	0,158	0,098	-0,467	0,955	-0,592
W0	W1	W2									
0,194	-0,057	0,603									

Individu 3 (P3):

Tabel 4.4 Representasi kromosom individu 3

V01	V02	V11	V12	V21	V22	V31	V32	V41	V42	V51	V52
0,959	0,667	-0,661	0,502	-0,035	-0,327	-0,440	-0,780	-0,171	-0,404	0,133	-0,707
W0	W1	W2									
0,125	0,717	-0,858									

Individu 4 (P4):

Tabel 4.5 Representasi kromosom individu 4

V01	V02	V11	V12	V21	V22	V31	V32	V41	V42	V51	V52
0,934	0,441	0,416	-0,542	-0,642	-0,963	0,260	0,092	0,426	0,401	-0,125	0,839
W0	W1	W2									
-0,724	-0,795	0,956									

Pada perhitungan secara manual ini akan digunakan 5 data latih dengan contoh pola pelatihan 5 bulan sebagai input dan bulan ke 6 sebagai output:

Tabel 4.6 Data latih dalam perhitungan manual

Pola	x1	x2	x3	x4	x5	y
1	0,847	0,564	0,528	0,613	0,323	0,314
2	0,580	0,528	0,613	0,323	0,314	0,191
3	0,535	0,613	0,323	0,314	0,191	0,162
4	0,641	0,323	0,314	0,191	0,162	0,140
5	0,565	0,245	0,304	0,185	0,121	0,112

4.4.2 Reproduksi

Proses reproduksi dilakukan untuk mendapatkan *offspring* dari individu terpilih pada generasi awal. Proses yang digunakan adalah proses *crossover* dan mutasi. Metode yang digunakan pada proses *crossover* ini adalah *one cut point*. Pada

metode ini ditentukan satu titik potong untuk kemudian dipasangkan dengan kromosom individu yang lain untuk membentuk keturunan yang baru.

Misalkan ditentukan pemilihan induk secara acak didapatkan induk adalah P1 dan P2, titik potongnya adalah titik ke-8 dari panjang kromosom, maka prosesnya adalah sebagai berikut:

Induk 1 *P1*

Tabel 4.7 Titik potong pada induk P1

V01	V02	V11	V12	V21	V22	V31	V32	V41	V42	V51	V52
0,328	0,427	0,996	-0,988	0,372	-0,448	0,923	-0,839	-0,208	-0,026	-0,480	0,843
W0	W1	W2									
0,081	-0,101	0,011									

Induk 2 *P3*

Tabel 4.8 Titik potong pada induk P2

V01	V02	V11	V12	V21	V22	V31	V32	V41	V42	V51	V52
-0,269	-0,371	0,482	-0,578	0,242	0,779	-0,853	0,158	0,098	-0,467	0,955	-0,592
W0	W1	W2									
0,194	-0,057	0,603									

Titik potong berada pada titik ke-8 sehingga didapatkan *C1* dan *C2* sebagai *offspring* atau keturunan.

C1

Tabel 4.9 Representasi kromosom hasil crossover C1

V01	V02	V11	V12	V21	V22	V31	V32	V41	V42	V51	V52
0,328	0,427	0,996	-0,988	0,372	-0,448	0,923	-0,839	0,098	-0,467	0,955	-0,592
W0	W1	W2									
0,194	-0,057	0,603									

C2

Tabel 4.10 Representasi kromosom hasil crossover C2

V01	V02	V11	V12	V21	V22	V31	V32	V41	V42	V51	V52
-0,269	-0,371	0,482	-0,578	0,242	0,779	-0,853	0,158	-0,208	-0,026	-0,480	0,843
W0	W1	W2									
0,081	-0,101	0,011									

Untuk proses mutasi digunakan metode *reciprocal exchange*. Metode ini akan menukar nilai dari 2 buah titik sehingga didapatkan kromosom *offspring* yang baru. Jika diandaikan individu yang terpilih sebagai induk untuk mutasi adalah P4,

dan titik yang digunakan untuk proses mutasi adalah titik 4 dan 12 maka didapatkan C3 sebagai berikut

C3

Tabel 4.11 Representasi kromosom hasil mutasi C3

V01	V02	V11	V12	V21	V22	V31	V32	V41	V42	V51	V52
0,959	0,667	-0,661	-0,707	-0,035	-0,327	-0,440	-0,780	-0,171	-0,404	0,133	0,502
W0	W1	W2									
0,125	0,717	-0,858									

4.4.3 Evaluasi

Proses evaluasi dilakukan dengan cara menghitung nilai MSE untuk masing-masing individu baik induk maupun anaknya, sehingga nantinya dapat dapat dilakukan proses seleksi terhadap individu yang ada. Seleksi dialkukan berdasarkan urutan nilai fitness yang didapatkan melalui Persamaan 4.1.

$$fitness = \frac{1}{MSE} \quad (4.1)$$

Hasil dari evaluasi masing-masing individu terdapat pada Tabel 4.11

Tabel 4.12 Proses evaluasi individu

Individu	MSE	*fitness*
P1	0,074	13,454
P2	0,113	8,872
P3	0,079	12,580
P4	0,045	22,135
P5	0,051	19,473
P6	0,190	5,274
P7	0,177	5,654
P8	0,060	16,710
P9	0,137	7,273
P10	0,126	7,967
C1	0,106	9,454
C2	0,077	12,912
C3	0,088	11,309

4.4.4 Seleksi

Proses seleksi dilakukan menggunakan metode elitism selection. Pemilihan metode ini dilakukan agar setiap individu dengan *fitness* terbaik adalah individu yang akan lolos ke tahap berikutnya. Sesuai dengan ukuran popSize=10, maka jumlah individu yang lolos ke tahap berikutnya adalah 10 individu dengan nilai *fitness* tertinggi. Individu-individu tersebut adalah:

Tabel 4.13 Proses seleksi individu

Individu(baru)	Individu(lama)	MSE	*fitness*
P1	P4	0,045	22,135
P2	P5	0,051	19,473
P3	P8	0,060	16,710
P4	P1	0,074	13,454
P5	C2	0,077	12,912
P6	P3	0,079	12,580
P7	C3	0,088	11,309
P8	C1	0,106	9,454
P9	P2	0,113	8,872
P10	P10	0,126	7,967

Jadi dalam proses seleksi ini didapatkan individu terbaik adalah P1. Representasi kromosom dari individu P1 ini yang nantinya akan terpilih menjadi bobot yang akan digunakan pada proses selanjutnya yaitu *feedforward*, *backpropagation* dan perubahan bobot. Setelah didapatkan individu terbaik, maka untuh tahapan berikutnya akan dilakukan perhitungan dengan metode *backpropagation* untuk melakukan update bobot dan bias. Representasi individu tebaik yaitu P1, merupakan bobot dan bias dari jaringan saraf yang akan digunakan sebagai inisialisasi bobot pada awal proses *feedforward*.

4.4.5 Perhitungan *feedforward*

Pada proses ini akan dilakukan perhitungan dengan bobot sesuai dengan optimasi bobot dan bias yang telah dilakukan oleh algoritme genetika. Langkahnya adalah dengan menghitung nilai z_in sebanyak unit yang ada dengan menggunakan Persamaan 2.5.

Z_in1 = 0,93 + 0,84*0,41 + 0,56*-0,64 + 0,52*0,26 + 0,61*0,42 = 1,283

Langkah berikutnya adalah menghitung nilai z1 menggunakan Persamaan 2.6.

Z1 = f(z_in1) = 0,783

Setelah itu dihitung nilai y_in menggunakan Persamaan 2.7.

Y_in = -0,72 + 0,783*-0.79 + 0,501*0.96 = -0,868

Selanjutnya dihitung nilai y’ dengan menggunakan aktivasi sigmoid bipolar. Aktifasi sigmoid bipolar digunakan karena menggunakan rentang antara -1 sampai dengan 1, dimana hal tersebut sesuai dengan bobot pada *input layer* yang berada pada rentang angka tersebut.

y’ = 0,296

Hasil perhitungan y’ secara lengkap ditunjukkan pada Tabel 4.13.

Tabel 4.14 Hasil perhitungan feedforward

Data	z_in1	z_in2	z1	z2	y_in	y'
1	1,283	0,003	0,783	0,501	-0,868	0,296
2	1,095	0,067	0,749	0,517	-0,826	0,305
3	0,958	-0,124	0,723	0,469	-0,850	0,299
4	1,137	0,024	0,757	0,506	-0,842	0,301
5	1,155	0,102	0,760	0,526	-0,826	0,304

4.4.6 Perhitungan *backpropagation*

Pada proses ini akan dilakukan perhitungan untuk menyesuaikan kesalahan hasil perhitungan dengan cara menghasilkan nilai yang dapat digunakan untuk menghasilkan bobot baru. Tahap awal proses ini adalah dengan mencari nilai δ_k menggunakan Persamaan 2.11.

δ_k = 0,018*0,296*(1-0,296) = 0,0038

Langkah selanjutnya adalah dengan menghitung nilai delta perubahan bobot w menggunakan Persamaan 2.12 dan bias menggunakan Persamaan 2.13.

Δw1 = 0,1*0,0038*0,783 = 0,0002

Hasil perhitungan Δw secara lengkap ditunjukkan pada Tabel 4.14

Tabel 4.15 Perhitungan *backpropagation* pada bobot w

Data	Δw1	Δw2	Δw0
1	0,000	0,000	0,000
2	-0,002	-0,001	-0,002
3	-0,002	-0,001	-0,003
4	-0,003	-0,002	-0,003
5	-0,003	-0,002	-0,004

Untuk menghitung nilai delta perubahan bobot v menggunakan Persamaan 2.16 dan bias menggunakan Persamaan 2.17.

Δv11 = 0,1*-0,0005*0,847 = -0,00004

Hasil perhitungan Δv secara lengkap ditunjukkan pada Tabel 4.15

Tabel 4.16 Perhitungan *backpropagation* pada bobot v

Data	Δv11	Δv12	Δv21	Δv22	Δvij	Δv52
1	0,0000	0,0001	0,0000	0,0001	...	0,0000
2	0,0002	-0,0003	0,0002	-0,0003	...	-0,0002
3	0,0002	-0,0004	0,0003	-0,0004	...	-0,0001
4	0,0003	-0,0005	0,0002	-0,0003	...	-0,0001
5	0,0003	-0,0006	0,0001	-0,0002	...	-0,0001

4.4.7 Perubahan Bobot

Hasil bobot baru setelah dilakukan perubahan hasil proses *backpropagation* adalah sebagai berikut:

Tabel 4.17 Bobot setelah dilakukan perubahan

V01	V02	V11	V12	V21	V22	V31	V32	V41	V42	V51	V52
0,935	0,440	0,417	-0,542	-0,641	-0,964	0,260	0,091	0,426	0,400	-0,125	0,839
w0	w1	w2									
-0,728	-0,798	0,954									

4.4.8 Perhitungan MSE dan *fitness*

Setelah didapatkan bobot baru dilakukan kembali proses *feedforward* untuk mendapatkan nilai MSE dengan menggunakan Persamaan 2.3.

MSE = 2,23017 / 50 =0,04460

Sedangkan untuk nilai *fitness* didapatkan dengan menggunakan Persamaan 2.4

fitness = 1/0,04460 = 22,4197

4.5 Perancangan Pengujian Algoritme Genetika

Pada tahapan ini akan dilakukan pengujian terhadap parameter algoritme genetika untuk mengetahui pengaruhnya terhadap kinerja sistem.

4.5.1 Pengujian Ukuran Populasi

Pada pengujian ini akan dilakukan penambahan ukuran populasi. Hal ini untuk mengetahui apakah semakin besar populasi akan berpengaruh terhadap nilai *fitness* yang dihasilkan. Perancangan pengujian ukuran populasi dilakukan mulai dengan ukuran populasi10 dan dilakukan setiap kelipatan 10 sampai dengan nilai ukuran populasi = 100 dan ditampilkan pada Tabel 4.17

Tabel 4.18 Perancangan pengujian ukurna populasi

Banyak populasi	Nilai *fitness* percobaan ke-										Rata2 fitness
	1	2	3	4	5	6	7	8	9	10	
10											
20											
…											
100											

4.5.2 Pengujian Jumlah Generasi

Pada pengujian ini akan dilakukan perubahan terhadap jumlah generasi. Hal ini bertujuan untuk mengetahui apakah perubahan tersebut akan berpengaruh terhadap nilai *fitness* yang dihasilkan. Perancangan pengujian ukuran generasi ditampilkan pada Tabel 4.18.

Tabel 4.19 Perancangan pengujian jumlah generasi

Jumlah generasi	Nilai *fitness* percobaan ke-										Rata2 fitness
	1	2	3	4	5	6	7	8	9	10	
10											
20											
…											
100											

4.5.3 Pengujian Kombinasi *Crossover* dan *Mutation Rate* (*Cr, Mr*)

Pada pengujian ini akan dilakukan perubahan nilai *Cr* dan *Mr*. Hal ini bertujuan untuk mengetahui apakah perubahan nilai tersebut akan berpengaruh terhadap nilai *fitness* yang dihasilkan. Perancangan pengujian kombinasi *Cr* dan *Mr* ditampilkan pada Tabel 4.19.

Tabel 4.20 Perancangan pengujian kombinasi Cr dan Mr

Cr	*Mr*	Nilai *fitness* percobaan ke-										Rata2 fitnes
		1	2	3	4	5	6	7	8	9	10	
0,9	0,1											
0,8	0,2											
0,7	0,3											
0,6	0,4											
0,5	0,5											
0,4	0,6											
0,3	0,7											
0,2	0,8											
0,1	0,9											

4.6 Perancangan Pengujian Perbandingan Data Latih dan Data Uji

Pada pengujian ini akan dilakukan pengujian dengan beberapa kombinasi prosentase jumlah data latih dan data uji. Hal ini bertujuan untuk mengetahui apakah perubahan jumlah data tersebut akan berpengaruh terhadap nilai MSE yang dihasilkan. Perancangan pengujian perbandingan data latih dan data uji ditampilkan pada Tabel 4.20

Tabel 4.21 Perancangan pengujian perbandingan data latih dan data uji

Data Latih	Data Uji	Nilai *MSE* percobaan ke-										Rata2 MSE
(%)	(%)	1	2	3	4	5	6	7	8	9	10	
90	10											
85	15											
80	20											
75	25											
70	30											
65	35											
60	40											
55	45											
50	50											

4.7 Perancangan Pengujian Metode *Backpropagation*

Pada tahapan ini akan dilakukan pengujian terhadap parameter jaringan saraf tiruan *backpropagation* untuk mengetahui pengaruhnya terhadap kinerja sistem.

4.7.1 Pengujian Jumlah Iterasi

Pada pengujian ini akan dilakukan pengujian terhadap jumlah iterasi untuk mengetahui apakah semakin banyak iterasi yang dilakukan akan berpengaruh terhada nilai MSE yang dihasilkan. Perancangan pengujian dilakukan mulai jumlah iterasiPerancangan pengujian jumlah iterasi ditampilkan pada Tabel 4.21.

Tabel 4.22 Perancangan pengujian jumlah iterasi

Jumlah Iterasi	Nilai *MSE* percobaan ke-										Rata2 MSE
	1	2	3	4	5	6	7	8	9	10	
50											
100											
150											
200											

4.7.2 Pengujian Nilai *Learning rate*

Pada Pengujian ini akan dilakukan pengujian terhadap nilai *learning rate* untuk mengetahui apakah dengan perubahan nilai *learning rate* yang dilakukan akan berpengaruh terhada nilai MSE yang dihasilkan. Perancangan pengujian dilakukan mulai learning rate 0,1 sampai dengan 0.9 Perancangan pengujian nilai *learning rate* ditampilkan pada Tabel 4.22

Tabel 4.23 Perancangan pengujian nilai *learning rate*

Learning rate	Nilai *MSE* percobaan ke-										Rata2 MSE
	1	2	3	4	5	6	7	8	9	10	
0,1											
0,2											
…											
0,9											

4.8 Perancangan Pengujian Perbandingan Metode *BP* dan GA-BP

Pada pengujian ini akan dilakukan pengujian dengan membandingkan nilai MSE yang dihasilkan dari hasil pengujian menggunakan metode *backpropagation* (BP) dan menggunakan gabungan metode algoritme genetika-*backpropagation* (GA-BP). Hal ini bertujuan untuk mengetahui apakah perbandingan kinerja kedua

metode tersebut. Perancangan pengujian perbandingan data latih dan data uji ditampilkan pada Tabel 4.23.

Tabel 4.24 Perancangan pengujian perbandingan BP dan GA-BP

Percobaan	Learning (MSE)		Testing (MSE)		Selisih	
	BP	GA-BP	BP	GA-BP	BP	GA-BP
1						
…						
10						

BAB 5 IMPLEMENTASI

Bab ini menjelaskan tentang implementasi sistem berdasarkan proses perancangan yang telah dibuat. Bab ini terdiri dari implementasi algoritme genetika dan implementasi jaringan saraf tiruan.

5.1 Implementasi Algoritme Genetika

5.1.1 Representasi kromosom individu

```
1     public void RepresentasiKromosom(){
2         populasi_awal = new double[popsize][panjangkromosom];
3         populasi = new double[popsize][panjangkromosom];
4         for (int i=0;i<popsize;i++){
5             for (int j=0;j<panjangkromosom;j++){
6                 populasi_awal[i][j] = (Math.random()*1,8 - 0.9);
7             }
8         }
9
10        for (int i=0; i<populasi_awal.length;i++){
11            for (int j=0; j<panjangkromosom;j++){
12                populasi[i][j]=populasi_awal[i][j];
13            }
14        }
15    }
```

Gambar 5.1 Kode program representasi kromosom

Penjelasan Gambar 5.1 adalah sebagai berikut:

1. Baris 2 melakukan inisialisasi jumlah populasi awal pada proses algoritme genetika.
2. Baris 3 melakukan inisialisasi jumlah populasi pada saat proses algoritme genetika berlangsung
3. Baris 4-8 melakukan inisialisasi kromosom individu ke dalam populasi awal secara acak sesuai dengan ukuran populasi dan panjang kromosom
4. Baris 10-15 memasukkan populasi awal ke dalam populasi untuk dilakukan proses algoritme genetika

5.1.2 Reproduksi

```
1         public void Reproduksi(){
2           int offspringC = (int)Math.round(popsize * cr);
3           int offspringM = (int)Math.round(popsize * mr);
4           childC = new double[offspringC][panjangkromosom];
5           childM = new double[offspringM][panjangkromosom];
6
7           for (int i = 0; i < offspringC; i++) {
8               randomParent = random.nextInt(range) + min;
9               for (int j = 0; j < panjangkromosom; j++) {
10                  this.childC[i][j] = populasi[randomParent-1][j];
11              }
12          }
13
14          for (int i = 0; i < offspringC; i++) {
15              cutPoint = random.nextInt(rangeC) + min;
16              randomParent = random.nextInt(range) + min;
17              for (int j = cutPoint-1; j < panjangkromosom; j++) {
18                  childC[i][j] = populasi[randomParent-1][j];
19              }
20          }
```

Gambar 5.2 Kode program *crossover*

Penjelasan Gambar 5.2 adalah sebagai berikut:

1. Baris 2 melakukan inisialisasi variable offspringC yang merupakan jumlah anak hasil proses *crossover*
2. Baris 3 melakukan inisialisasi variable offspringM yang merupakan jumlah anak hasil proses mutasi
3. Baris 4-5 melakukan inisialisasi anak hasil proses reproduksi
4. Baris 7-12 melakukan perulangan untuk menentukan individu yang berperan sebagai induk pada proses *crossover* yang dipilih secara acak dari populasi yang ada
5. Baris 14-20 melakukan perulangan untuk menentukan titik yang menjadi titik potong dalam proses *crossover.*

```
1           for (int i = 0; i < offspringM; i++) {
2               randomParent = random.nextInt(range) + min;
3
4               for (int j = 0; j < panjangkromosom; j++) {
5                   tempOffspringM[i][j]    =    populasi[randomParent-
6   1][j];
7               }
8           }
9
10          for (int i = 0; i < offspringM; i++) {
11              exchangePoint1 = random.nextInt(rangeM) + minM;
12              exchangePoint2 = random.nextInt(rangeM) + minM;
13
14              if (exchangePoint2 == exchangePoint1) {
15                  exchangePoint2 = random.nextInt(rangeM) + minM;
16              }
17
18              for (int j = 0; j < panjangkromosom; j++) {
19                  if (j == (exchangePoint1-1)) {
20                      this.childM[i][j]                                =
21  tempOffspringM[i][exchangePoint2-1];
22                  } else if (j == (exchangePoint2-1)) {
23                      this.childM[i][j]                                =
24  tempOffspringM[i][exchangePoint1-1];
25                  } else{
26                      this.childM[i][j] = tempOffspringM[i][j];
                    }
                }
            }
```

Gambar 5.3 Kode program proses mutasi

Penjelasan Gambar 5.3 adalah sebagai berikut:

1. Baris 1-7 melakukan perulang untuk menentukan individu mana yang terpilih menjadi induk dalam proses mutasi
2. Baris 10-11 melakukan proses penentuan titik untuk saling menukar pada kromosom yang dilakukan secara acak pada induk yang terpilih
3. Baris 13-14 melakukan cek kondisi jika titik yang terpilih untuk proses saling tukar sama, maka dilakukan pengacakan lagi pada titik kedua
4. Baris 17-26 melakukan proses penukaran gen menggunakan dua titik yang telah ditentukan untuk menghasilkan anak hasil proses mutasi

5.1.3 Evaluasi

```
1       public void Evaluasi() throws IOException, BiffException{
2           fitness = new double[himpunanIndividu.length];
3           for (int e=0;e<himpunanIndividu.length;e++){
4               InisialisasiBobotGA(e);
5               for (int epoch = 0; epoch < 1; epoch++){
6               double sum_error=0.0;
7               for( int data=0; data < max_data; data++){
8                   for(int i=0; i< neuron_input; i++){
9                       row_input[i] = file_input[data][i];
10                  }
11                  target = file_output[data][0];
12                  PropagasiMaju();
13                  sum_error = sum_error+(Math.pow(error, 2));
14              }
15              mse[epoch] = (sum_error/max_data);
16              fitness[e] = 1/mse[epoch];
17              }
18          }
19      }
```

Gambar 5.4 Kode program proses evaluasi

Penjelasan Gambar 5.4 adalah sebagai berikut:

1. Baris 2 melakukan inisialisasi variabel *fitness* yang akan digunakan sebagai dasar evaluasi
2. Baris 3-4 melakukan insialisasi bobot dari kromosom masing-masing individu
3. Baris 6 melakukan inisialisasi variable *sum_error*
4. Baris 8-10 melakukan perulangan memasukkan data uji ke dalam variable *row_input*
5. Baris 11 melakukan pengambilan nilai dari file_output untuk dimasukkan dalam variable target
6. Baris 12 memanggil proses *feedforward*
7. Baris 13 melakukan perhitungan *sum_error* selama perulangan dilakukan.
8. Baris 15 melakukan perhitungan nilai MSE
9. Baris 16 melakukan perhitungan *fitness* yang dihasilkan

5.1.4 Seleksi

```
1         public void seleksi(){
2          double temp;
3          int index;
4          int[] tempIndex = new int[this.fitness.length];
5          for (int i = 0; i < tempIndex.length; i++) {
6              tempIndex[i] = i;
7          }
8
9          boolean selesai = true;
10         while(selesai){
11             selesai = false;
12             for (int i = 1; i < this.fitness.length; i++) {
13                 if (fitness[i-1] < fitness[i]) {
14                     temp = fitness[i-1];
15                     index = tempIndex[i-1];
16                     fitness[i-1] = fitness[i];
17                     tempIndex[i-1] = tempIndex[i];
18                     fitness[i] = temp;
19                     tempIndex[i] = index;
20                     selesai = true;
21                 }
22             }
23         }
```

Gambar 5.5 Kode program proses seleksi

Penjelasan Gambar 5.5 adalah sebagai berikut:

1. Baris 4 melakukan inisialisasi variable tempIndex yang akan digunakan sebagai variable penyimpan untuk pengurutan *fitness*
2. Baris 10-22 melakukan pengurutan mulai dari *fitness* terbesar sampai *fitness* terkecil

5.2 Implementasi Jaringan Saraf Tiruan

5.2.1 Inisialisasi Bobot Awal

Tahap ini adalah awal dari jaringan saraf tiruan. Penentuan bobot awal merupakan hasil dari optimasi yang dilakukan oleh algoritme genetika.

```
1       public void BobotGA2BP(){
2           int count =0;
3           int batasvij= panjangkromosom-(neuron_hidden+1);
4           for(int i = 0; i<= neuron_input; i++){
5               for(int j=0; j< neuron_hidden; j++){
6                   if(count>batasvij) break;
7                   vij[i][j] = individuTerbaik[count];
8                   count++;
9               }    }
10          int countw = panjangkromosom-(neuron_hidden+1);
11          for(int j = 0; j<= neuron_hidden; j++){
12              for(int k=0; k< neuron_output; k++){
13                  if(countw==panjangkromosom) break;
14                  wjk[j][k] = individuTerbaik[countw];
15                  countw++;
16          }} }
```

Gambar 5.6 Kode program inisialisasi bobot awal

Penjelasan Gambar 5.6 adalah sebagai berikut:

1. Baris 2-6 merupakan inisialisasi bobot awal yang terletak di antara input *layer* dan *hidden layer*. Bobot pada bagian ini disebut dengan Vij. Perulangan dilakukan sesuai jumlah neuron pada *hidden layer* dan input *layer* untuk mendapatkan keseluruhan bobot Vij secara acak.
2. Baris 8-13 merupakan inisialisasi bobot awal yang terletak diantara *hidden layer* dan output *layer*. Bobot pada bagian ini disebut dengan Wjk. Perulangan dilakukan sesuai jumlah neuron pada output *layer* dan *hidden layer* untuk mendapatkan keseluruhan bobot Wjk secara acak

5.2.2 Proses *Feedforward*

```
1     public void PropagasiMaju(){
2         double z_in;
3         double y_in;
4         for(int i=0; i<neuron_hidden; i++){
5             for(int j=0; j < neuron_input; j++){
6                 z_in = z_in +(row_input[j]*vij[j][i]);
7             }
8             z_in = z_in+ vij[neuron_input][i];
9             zedd[i] = FungsiAktivasi(z_in);
10        }
11        for(int i=0; i<neuron_hidden; i++){
12            y_in = y_in + (zedd[i]*wjk[i][0]);
13        }
14        y_in = y_in + wjk[neuron_hidden][0];
15        y_output = FungsiAktivasi(y_in);
16    }
```

Gambar 5.7 Kode program tahap *feedforward*

Penjelasan Gambar 5.7 adalah sebagai berikut:

1. Baris 2-3 merupakan inisialisasi variable *z_in* dan *y_in*.
2. Baris 4-10 merupakan perulangan yang dilakukan berdasarkan jumlah neuron pada input *layer* dan jumlah neuron pada *hidden layer* untuk mendapatkan nilai *z_in* yang merupakan nilai keluaran dari proses perkalian antara nilai neuron pada input *layer* dengan bobot Vij. Selanjutnya akan dilakukan fungsi aktifasi pada nilai *z_in* sehingga menghasilkan nilai *zedd*.
3. Baris 11-16 merupakan perulangan yang dilakukan berdasarkan jumlah neuron pada *hidden layer* dan jumlah neuron pada output *layer* untuk mendapatkan nilai *y_in* yang merupakan nilai keluaran dari proses perkalian antara nilai neuron pada input *layer* dengan bobot Wjk. Selanjutnya akan dilakukan fungsi aktifasi pada nilai yin sehingga menghasilkan nilai *y_output*.

5.2.3 Proses *Backpropagation*

```
1   public void PropagasiMundur(){
2     error = (target-y_output);
3     faktor_error_output=(error*y_output*(1-y_output));
4     for(int i=0; i<neuron_hidden;i++){
5        deltawjk[i] = error*zedd[i]*learning_rate;
6     }
7     deltawjk[neuron_hidden] = error*learning_rate; //delta bias
8     for(int i=0; i<neuron_hidden; i++){
9         faktor_error_hidden[i] =error*wjk[i][0]*zedd[i]*(1-zedd[i]);
10    }
11    for(int i=0; i<neuron_hidden; i++){
12       for(int j=0; j<neuron_input; j++){
13          deltavij[j][i]=
14  faktor_error_hidden[i]*zedd[j]*learning_rate;
15       }
16       deltavij[neuron_input][i]=
17  faktor_error_hidden[i]*learning_rate;
18          }
19      }
```

Gambar 5.8 Kode program tahap *backpropagation*

Penjelasan Gambar 5.8 adalah sebagai berikut:

1. Baris 2 merupakan perhitungan untuk mencari selisih antara hasil peramalan dan data uji.
2. Baris 3 adalah perhitungan untuk mendapatkan factor error output.
3. Baris 4-6 merupakan perulangan untuk mendapatkan delta bobot antara *hidden layer* dan output *layer*.
4. Baris 7 merupakan perhitungan untuk mencari nilai delta bias antara *hidden layer* dan output *layer*
5. Baris 8-10 merupakan perulangan untuk mendapatkan faktor *error* antara input *layer* dan *hidden layer*.
6. Baris 11-15 merupakan perulangan untuk mendapatkan nilai delta bobot antara input *layer* dan *hidden layer*.
7. Baris 16 merupakan perhitungan untuk mencari nilai delta bias antara input *layer* dan *hidden layer*.

5.2.4 Proses Update Bobot

```
1    public void UpdateBobot(){
2      for(int i=0; i<neuron_hidden; i++){
3          for(int j=0; j<neuron_input; j++){
4              vij[j][i] =  vij[j][i] + deltavij[j][i];
5          }
6
7    vij[neuron_input][i]=vij[neuron_input][i]+deltavij[neuron_input][i];
8        }
9        for(int i=0; i<neuron_output; i++){
10           for(int j=0; j<neuron_hidden;j++){
11                wjk[j][i] =  wjk[j][i] + deltawjk[j];
12            }
13   wjk[neuron_hidden][i]=wjk[neuron_hidden][i]+deltawjk[neuron_hidden];
14           }
15       }
```

Gambar 5.9 Kode program tahap perubahan bobot

Penjelasan Gambar 5.9 adalah sebagai berikut:

1. Baris 2-5 merupakan perulangan untuk melakukan perubahan bobot antara input *layer* dan *hidden layer*.
2. Baris 6 merupakan perhitungan untuk melakukan perubahan bias antara input *layer* dan *hidden layer*.
3. Baris 9-12 merupakan perulangan untuk melakukan perubahan bobot antara *hidden layer* dan output *layer*.
4. Baris 13 merupakan perhitungan untuk melakukan perubahan bias antara *hidden layer* dan output *layer*.

BAB 6 PENGUJIAN DAN ANALISIS

Bab ini membahas mengenai hasil dari pengujian yang dirancang pada bab sebelumnya. Terdapat 3 pengujian yang dilakukan, yaitu terhadap algoritme genetika, metode *backpropagation*, dan perbandingan hasil uji antara kedua metode tersebut.

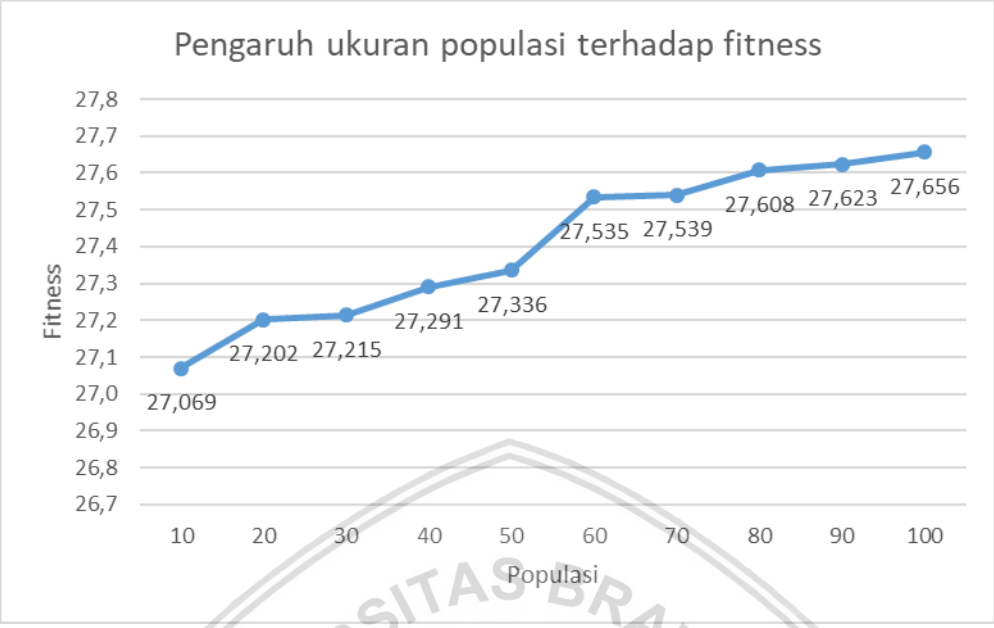
6.1 Pengujian Algoritme Genetika

6.1.1 Pengujian Ukuran Populasi

Pengujian ini dilakukan untuk mengetahui pengaruh ukuran populasi terhadap nilai *fitness* yang dihasilkan. Pada pengujian ukuran populasi ini digunakan parameter banyak generasi 10, Cr adalah 0,5 dan Mr adalah 0,5. Pengujian dilakukan mulai ukuran populasi 10 sampai dengan 100. Hasil pengujian nilai rata-rata *fitness* pada setiap percobaan dapat dilihat pada Lampiran. Hasil pengujian pengaruh ukuran populasi terhadap rata-rata *fitness* ditampilkan pada Tabel 6.1.

Tabel 6.1 Pengujian ukuran populasi

Ukuran populasi	Rata-rata MSE	*Fitness*
10	0,0369	27,069
20	0,0368	27,202
30	0,0367	27,215
40	0,0366	27,291
50	0,0366	27,336
60	0,0363	27,535
70	0,0363	27,539
80	0,0362	27,608
90	0,0362	27,623
100	0,0362	27,656



Gambar 6.1 Grafik hasil pengujian ukuran populasi

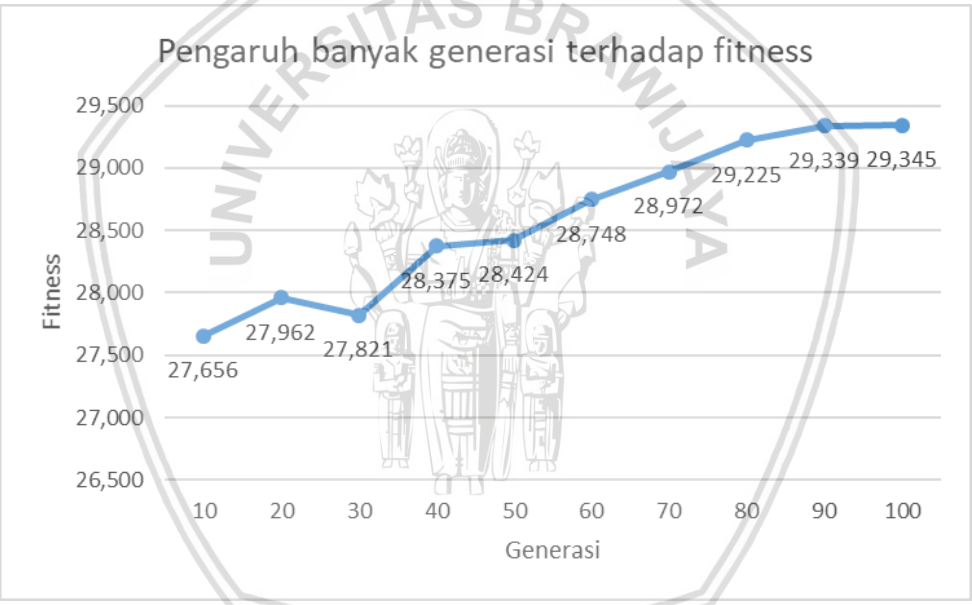
Dari hasil uji coba ukuran populasi yang ditampilkan seperti pada Gambar 6.1, dapat diambil kesimpulan bahwa semakin besar ukuran populasi semakin besar pula nilai *fitness* yang dihasilkan. Ukuran populasi yang besar mencerminkan semakin banyak individu yang dapat ditampung. Semakin banyak individu dalam suatu populasi akan lebih menungkinkan mendapatkan nilai *fitness* dari masing-masing individu yang lebih beragam. Semakin besar ukuran populasi akan memperbesar peluang terjadinya reproduksi baik melalui *crossover* maupun mutasi sesuai dengan besar parameter Mr dan Cr. Semakin sering terjadi reproduksi maka semakin besar pula peluang untuk mendapatkan nilai *fitness* yang lebih baik dibanding induknya. Nilai *fitness* terbaik didapatkan pada ukuran populasi 100 sebesar 27,65.

6.1.2 Pengujian Banyak Generasi

Pengujian ini dilakukan untuk mengetahui pengaruh banyak generasi terhadap nilai *fitness* yang dihasilkan. Parameter yang digunakan pada pengujian ini yaitu ukuran populasi 100, Cr adalah 0,5 dan Mr adalah 0,5. Pengujian dilakukan mulai banyak generasi 10 sampai dengan 100. Hasil pengujian nilai rata-rata *fitness* pada setiap percobaan dapat dilihat pada Lampiran. Hasil pengujian pengaruh banyak generasi terhadap *fitness* ditampilkan pada Tabel 6.2.

Tabel 6.2 Pengujian banyak generasi

Banyak Generasi	Rata-rata MSE	*Fitness*
10	0,036	27,656
20	0,036	27,962
30	0,036	27,821
40	0,035	28,375
50	0,035	28,424
60	0,035	28,748
70	0,035	28,972
80	0,034	29,225
90	0,034	29,339
100	0,034	29,345



Gambar 6.2 Grafik hasil pengujian banyak generasi

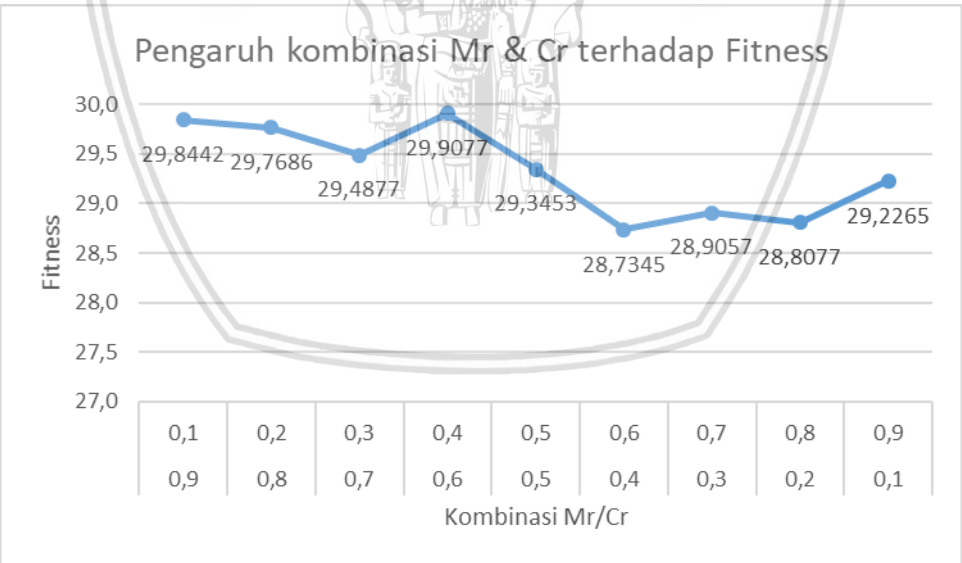
Dari hasil uji coba banyak generasi seperti pada Gambar 6.2, dapat disimpulkan bahwa semakin banyak generasi maka cenderung mendapatkan nilai *fitness* yang semakin baik. Semakin banyak generasi semakin banyak pula individu yang dihasilkan dan semakin besar pula peluang mendapatkan individu yang memiliki nilai *fitness* yang lebih tinggi. Semakin banyak generasi menunjukkan iterasi yang dilakukan saat proses optimasi menggunakan algoritme genetika, yaitu reproduksi, evaluasi dan seleksi. Semakin sering proses algoritme genetika dilakukan terdapat peluang akan muncul individu baru sebagai individu terbaik dengan nilai *fitness* terbaik. Nilai *fitness* terbaik didapatkan pada banyak generasi 100 sebesar 29,345.

6.1.3 Pengujian Kombinasi *Cr* dan *Mr*

Pengujian ini dilakukan untuk mengetahui kombinasi Cr dan Mr untuk mendapatkan nilai *fitness* terbaik. Parameter yang digunakan pada pengujian ini yaitu ukuran populasi 100 dan banyak generasi 90. Pengujian dilakukan sebanyak 10 kali. Hasil pengujian nilai rata-rata *fitness* pada setiap percobaan dapat dilihat pada Lampiran. Hasil pengujian pengaruh kombinasi Cr dan Mr terhadap rata-rata *fitness* ditampilkan pada Tabel 6.3.

Tabel 6.3 Pengujian kombinasi Cr dan Mr

Cr	*Mr*	Rata-rata MSE	*Fitness*
0,9	0,1	0,0335	29,8442
0,8	0,2	0,0336	29,7686
0,7	0,3	0,0339	29,4877
0,6	0,4	0,0334	29,9077
0,5	0,5	0,0341	29,3453
0,4	0,6	0,0348	28,7345
0,3	0,7	0,0346	28,9057
0,2	0,8	0,0347	28,8077
0,1	0,9	0,0342	29,2265



Gambar 6.3 Grafik hasil pengujian kombinasi Cr dan Mr

Dari hasil pada Gambar 6.3 dapat disimpulkan bahwa nilai Cr yang lebih besar dibandingkan Mr akan menghasilkan nilai *fitness* yang lebih baik. Nilai Cr dan Mr merupakan parameter yang menentukan banyaknya jumlah individu yang lahir karena proses reproduksi. Terlihat bahwa pada saat nilai Cr lebih besar daripada Mr, nilai *fitness* yang dihasilkan cenderung lebih besar bila dibandingkan saat nilai Mr lebih besar daripada Cr. Hal tersebut menunjukkan bahwa proses *crossover*

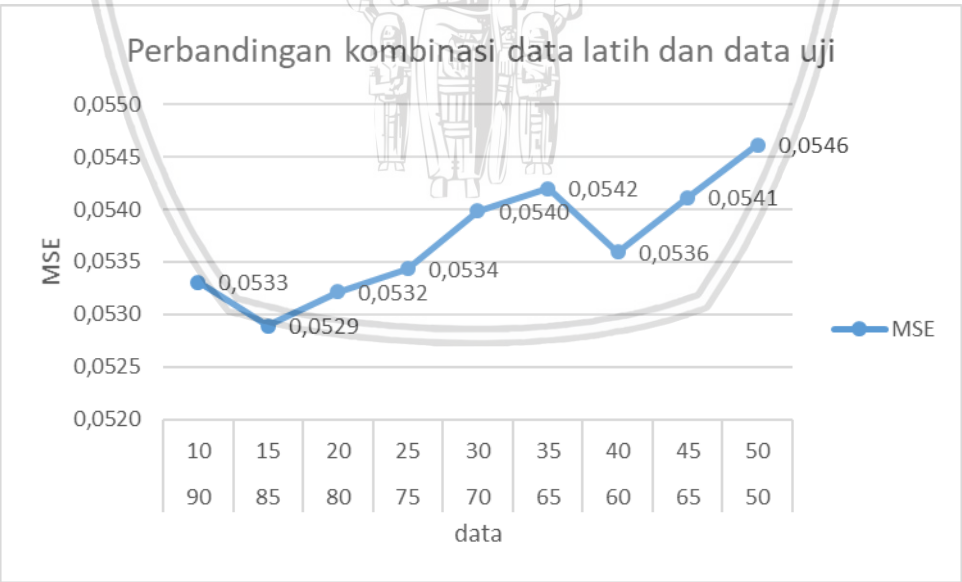
lebih berperan dalam menghasilkan individu-individu dengan nilai *fitness* yang lebih baik dibandingkan proses mutasi. Nilai *fitness* terbaik didapatkan pada kombinasi Cr 0,6 dan Mr 0,4 yaitu sebesar 29,9077.

6.2 Pengujian Perbandingan Data Latih dan Data Uji

Pengujian ini dilakukan untuk mengetahui komposisi terbaik dari jumlah data latih dan data uji yang digunakan.

Tabel 6.4 Pengujian perbandingan data latih dan data uji

Data Latih (%)	*Data Uji* (%)	Rata-rata MSE
90	10	0,0533
85	15	0,0529
80	20	0,0532
75	25	0,0534
70	30	0,0540
65	35	0,0542
60	40	0,0536
65	45	0,0541
50	50	0,0546



Gambar 6.4 Grafik pengujian kombinasi data latih dan data uji

Dari hasil pada Gambar 6.4 didapatkan bahwa nilai rata-rata MSE terendah didapatkan pada saat rasio data latih:data uji adalah 85:15 dengan nilai MSE terbaik adalah 0,0529. Penggunaan data yang terlalu sedikit cenderung menghasilkan nilai rata-rata MSE yang tinggi, hal tersebut karena terlalu seditkitnya data yang digunakan untuk pembelajaran oleh sistem. Namun data

yang banyak juga bukan jaminan bahwa akan didapatkan nilai rata-rata MSE yang paling kecil. Hal tersebut disebabkan adanya kemungkinan jarak antara data tidak membentuk pola yang baik.

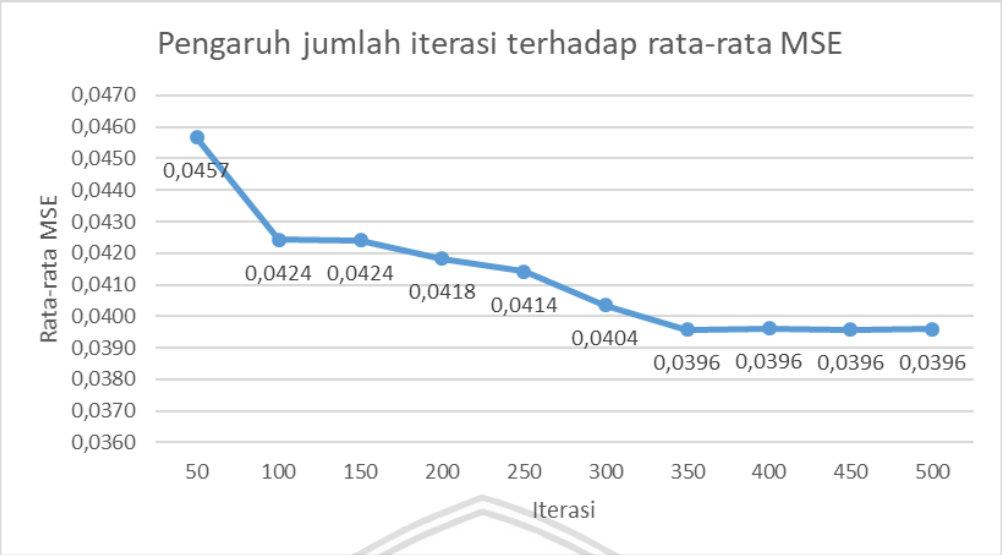
6.3 Pengujian Metode *Backpropagation*

6.3.1 Pengujian Jumlah Iterasi

Pengujian ini dilakukan untuk mendapatkan nilai MSE terbaik yaitu nilai MSE yang paling rendah. Iterasi yang dilakukan sebanyak 50 sampai dengan 500 iterasi. Untuk masing-masing iterasi dilakukan sebanyak 10 kali. Nilai *learning rate* ditetapkan 0,5. Hasil pengujian nilai MSE pada setiap percobaan dapat dilihat pada Lampiran. Hasil pengujian pengaruh jumlah iterasi terhadap rata-rata MSE dapat dilihat pada Tabel 6.5.

Tabel 6.5 Pengujian pengaruh jumlah iterasi

Jumlah Iterasi	Rata-rata MSE
50	0,0457
100	0,0424
150	0,0424
200	0,0418
250	0,0414
300	0,0404
350	0,0396
400	0,0396
450	0,0396
500	0,0396



Gambar 6.5 Grafik hasil pengujian jumlah iterasi

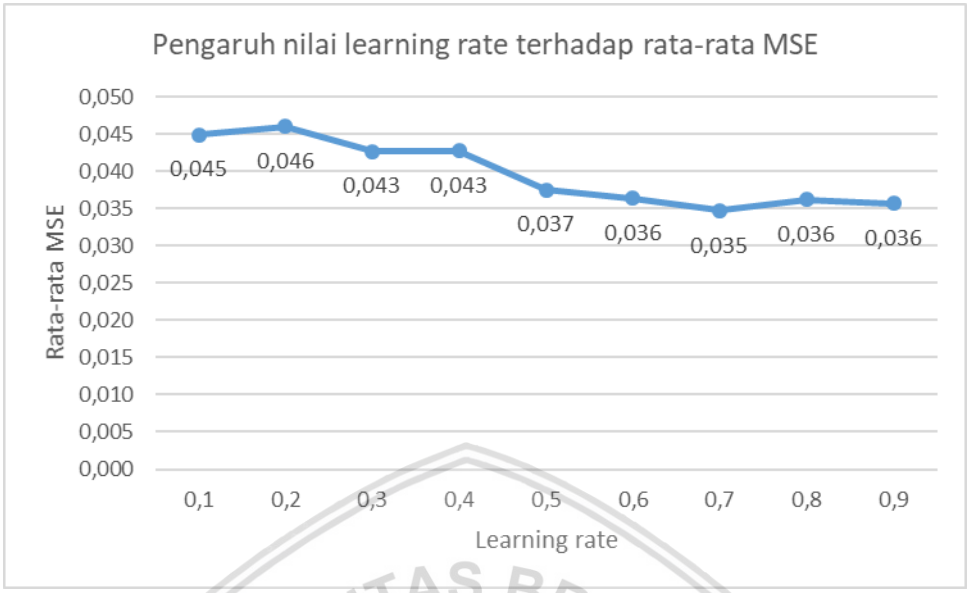
Dari hasil uji coba jumlah iterasi yang ditampilan pada Gambar 6.5 dapat disimpulkan bahwa semakin banyak iterasi yang dilakukan akan menghasilkan nilai rata-rata MSE yang semakin rendah. Semakin banyak jumlah iterasi yang dilakukan maka semakin banyak pula peluang untuk mendapatkan nilai bobot yang lebih optimal, sehingga dapat menghasilkan nilai MSE yang lebih rendah. Pada iterasi ke 350 didapatkan nilai MSE terbaik yaitu 0,036. Setelah itu walaupun iterasi bertambah namun nilai MSE tidak menunjukkan perubahan yang signifikan.

6.3.2 Pengujian Nilai *Learning rate*

Pengujian ini dilakukan untuk mengetahui pengaruh nilai *learning rate* pada nilai MSE yang dihasilkan dari implementasi sistem. Percobaan dilakukan mulai nilai *learning rate* 0,1 sampai dengan 0,9. Jumlah iterasi yang digunakan adalah 500. Hasil pengujian nilai MSE pada setiap percobaan dapat dilihat pada Lampiran. Hasil pengujian pengaruh parameter nilai *learning rate* terhadap nilai MSE dapat dilihat pada Tabel 6.6.

Tabel 6.6 Pengujian nilai *learning rate*

Learning rate	Rata-rata MSE
0,1	0,045
0,2	0,046
0,3	0,043
0,4	0,043
0,5	0,037
0,6	0,036
0,7	0,035
0,8	0,036
0,9	0,036



Gambar 6.6 Grafik hasil pengujian *learning rate*

Berdasarkan Gambar 6.6 dapat disimpulkan semakin besar nilai *learning rate* cenderung menghasilkan nilai rata-rata MSE yang lebih rendah. Pengecualian terjadi pada nilai *learning rate* 0,4 yang menghasilkan nilai MSE terkecil pada saat pengujian. Hal ini disebabkan pada saat inisialisasi bobot yang dilakukan secara acak, nilai bobotnya sudah mendekati nilai optimal sehingga dengan perubahan bobot sesuai *learning rate* 0,4 sudah menghasilkan nilai MSE yang rendah. Pada dasarnya nilai *learning rate* akan mempercepat perubahan bobot jaringan untuk mencapai nilai MSE yang lebih rendah, namun bukan berarti bahwa semakin besar nilai *learning rate* akan selalu menghasilkan nilai MSE terendah. Hal tersebut dikarenakan semakin besar nilai *learning rate* akan memperbesar pula perubahan bobot, sehingga ada kemungkinan nilai optimum bobot jaringan akan terlewat karena perubahan bobot yang besar tersebut. Nilai rata-rata MSE terbaik terdapat pada nilai *learning rate* 0,4 yaitu sebesar 0,035.

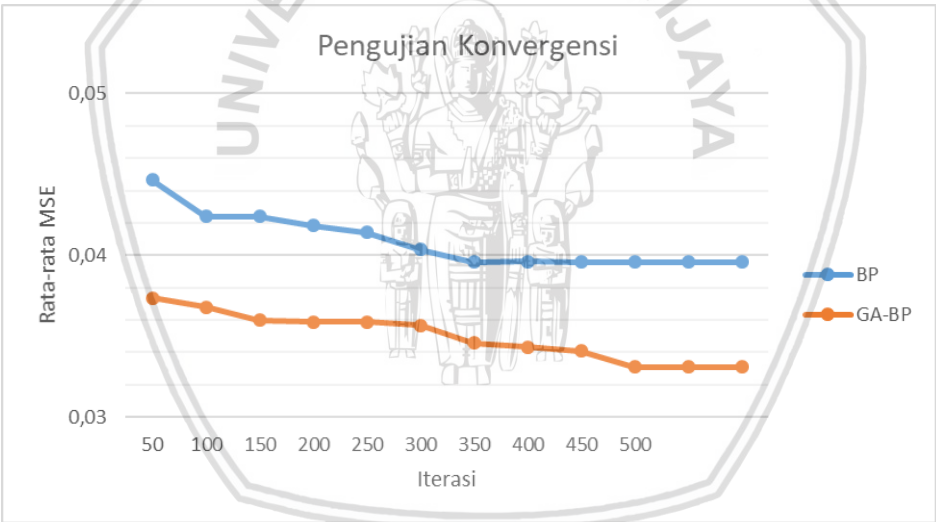
6.4 Perbandingan Metode BP dan GA-BP

6.4.1 Pengujian Konvergensi

Pengujian ini dilakukan untuk mengetahui berapa banyak iterasi yang dibutuhkan hingga menghasilkan MSE terkecil. Pengujian ini menggunakan maksimum terasi sebesar 600, learning rate sebesar 0.5. Pengujian akan dilakukan sebanyak 10 kali percobaan kemudian dilihat pada iterasi keberapa MSE mencapai konvergen, yaitu situasi dimana nilai MSE sedikit atau tidak menjadi lebih baik secara signifikan jika dilanjutkan ke iterasi selanjutnya. Hasil pengujian konvergensi ditunjukkan pada Tabel 6.7.

Tabel 6.7 Pengujian Konvergensi

Iterasi	MSE	
	BP	GA-BP
50	0,045	0,037
100	0,042	0,037
150	0,042	0,036
200	0,042	0,036
250	0,041	0,036
300	0,040	0,036
350	0,040	0,035
400	0,040	0,034
450	0,040	0,034
500	0,040	0,033
550	0,040	0,033
600	0,040	0,033



Gambar 6.7 Grafik hasil pengujian konvergensi

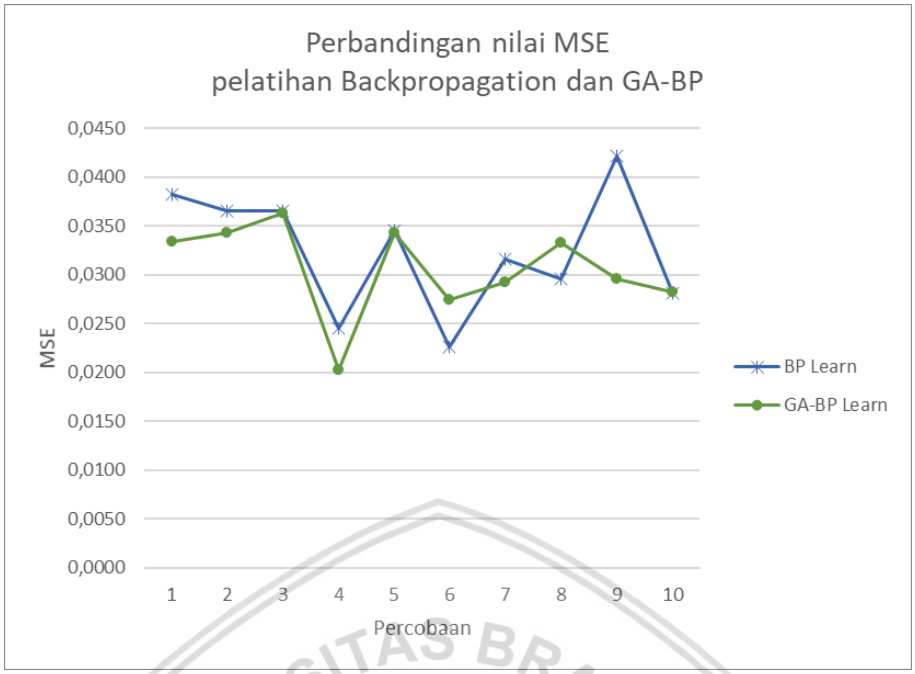
Berdasarkan Gambar 6.7 dapat disimpulkan bahwa optimasi yang dilakukan oleh algoritme genetika menyebabkan iterasi pada saat terjadi konvergensi menjadi meningkat yaitu pada iterasi ke 500, sedangkan apabila hanya menggunakan backpropagation saja konvergensi dapat dicapai pada iterasi ke 300. Hal ini menunjukkan bahwa algoritme genetika memperbaiki kelemahan backpropagation yang seringkali terjebak pada local minimum. Penggunaan algoritme genetika akan membuat hasil akan menghindari local minimum dan mendekati global minimum.

6.4.2 Pengujian Hasil Peramalan

Pengujian ini dilakukan untuk mengetahui perbandingan kinerja motode *backpropagation* dan GA-BP dalam melakukan prediksi debit bendungan. Pada pengujian ini dilakukan 2 tahap yaitu tahap pelatihan/training dan pengujian/testing. Pada tahap pelatihan baik metode *backpropagation* maupun GA-BP akan dilatih menggunakan data latih dari data time series debit bendungan. Setelah didapatkan pembelajaran, maka bobot hasil dari proses pembelajaran tersebut akan digunakan untuk melakukan pengujian melalui data uji. Parameter yang digunakan pada pengujian ini adalah parameter optimal yang didapatkan dari pengujian sebelumnya yaitu jumlah iterasi=500, *learning rate*=0,7 untuk parameter *backpropagation* dan ukuran populasi=100, banyak generasi=100, Cr=0,6 dan Mr=0,4 untuk parameter algoritme genetika. Hasil dari pengujian ditampilkan pada Tabel 6.8.

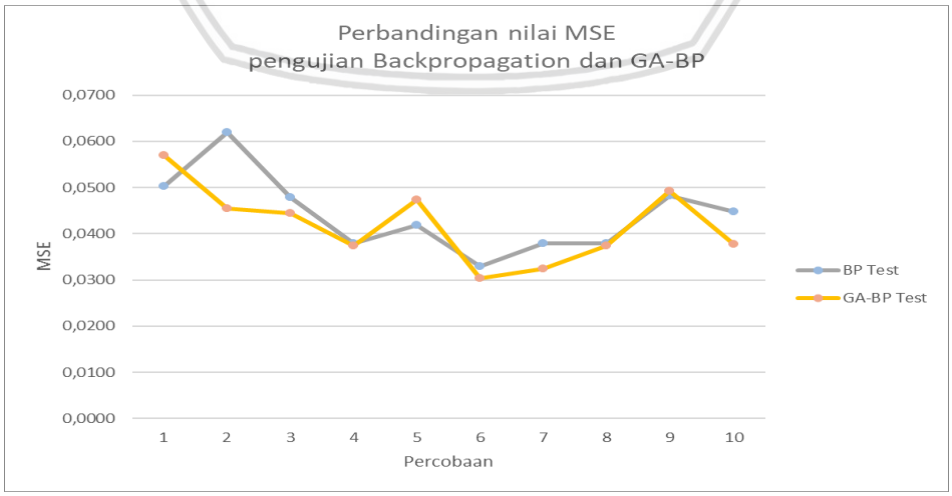
Tabel 6.8 Pengujian perbandingan *BP* dan GA-BP

Percobaan	Learning (MSE)		Testing (MSE)		Selisih	
	BP	GA-BP	BP	GA-BP	BP	GA-BP
1	0,03821	0,03334	0,05025	0,05697	0,01204	0,02363
2	0,03655	0,03427	0,06192	0,04539	0,02537	0,01112
3	0,03655	0,03627	0,04792	0,04439	0,01137	0,00812
4	0,02455	0,02027	0,03792	0,03739	0,01337	0,01712
5	0,03455	0,03427	0,04192	0,04739	0,00737	0,01312
6	0,02255	0,02739	0,03292	0,03039	0,01037	0,00300
7	0,03155	0,02927	0,03792	0,03239	0,00637	0,00312
8	0,02955	0,03327	0,03792	0,03739	0,00837	0,00412
9	0,04213	0,02960	0,04828	0,04931	0,00616	0,01971
10	0,02809	0,02823	0,04477	0,03782	0,01668	0,00959
Rata-rata	0,03243	0,03062	0,04417	0,04188	0,01175	0,01126



Gambar 6.8 Grafik perbandingan nilai pelatihan BP dan GA-BP

Dari hasil pada Gambar 6.8 terlihat bahwa nilai MSE hasil pelatihan dengan menggunakan GA-BP cenderung lebih rendah apabila dibandingkan dengan metode *backpropagation*. Hal ini disebabkan pada pada proses algoritme genetika, bobot yang dibangkitkan secara acak, akan dioptimasi terlebih dahulu dan dilakukan proses reproduksi untuk mendapatkan individu dengan *fitness* terbaik yang nantinya akan dijadikan sebagai bobot awal untuk proses jaringan saraf tiruan, sehingga bobot awal tersebut telah melalui serangkaian pengujian untuk memastikan bobot awal adalah bobot yang memiliki nilai MSE terendah. Hal tersebut menunjukkan bahwa bobot hasil dari GA-BP menghasilkan nilai peramalan yang lebih baik pada saat dilakukan pengujian dengan data uji yang ada.



Gambar 6.9 Grafik perbandingan pengujian BP dan GA-BP

Dari hasil pada Gambar 6.9 terlihat bahwa hasil pengujian dari GA-BP memiliki nilai MSE yang lebih baik dibandingkan dengan metode *backpropagation*. Nilai MSE yang lebih rendah menunjukkan hasil yang lebih baik.

Dari hasil pengujian GA-BP dengan nilai terbaik selanjutnya akan dilakukan denormalisasi data untuk mengetahui bagaimana kondisi aktual debit bendungan hasil peramalan apa bila dibandingkan dengan kondisi aktual pada data uji. Hasil pengujian secara lengkap ditampilkan dalam Lampiran. Hasil denormalisasi secara singkat ditampilkan pada Tabel 6.9.

Tabel 6.9 Hasil peramalan

No	Normalisasi		Denormalisasi		
	Peramalan	Aktual	Peramalan	Aktual	Selisih
1	0,4037	0,4441	80,8794	88,3527	7,4732
2	0,6411	0,6852	124,7302	132,8740	8,1438
3	0,7235	0,8022	139,9337	154,4699	14,5362
4	0,9305	0,8949	178,1606	171,5865	6,5741
5	0,9242	0,9000	176,9989	172,5313	4,4676
6	0,9031	0,8397	173,1034	161,4051	11,6983
7	0,8089	0,7593	155,7083	146,5482	9,1600
8	0,6202	0,5681	120,8569	111,2380	9,6188
9	0,4889	0,4327	96,6245	86,2483	10,3762
10	0,3928	0,2841	78,8757	58,8095	20,0663
11	0,231563	0,298424	49,100542	61,44681	12,3463
12	0,2485	0,2774	52,2342	57,5578	5,3237
13	0,322372	0,242119	65,868954	51,04971	14,8192
14	0,183363	0,17382	40,200131	38,43788	1,7623
15	0,409796	0,441752	82,012305	87,91312	5,9008
16	0,672396	0,676103	130,50293	131,1874	0,6845
				Rata-rata selisih	8,9345

Dari hasil proses denormalisasi didapatkan hasil peramalan yang dilakukan menggunakan GA-BP mempunyai selisih rata-rata debit bendungan sebesar 8,9503 m3/detik untuk setiap bulannya.

BAB 7 PENUTUP

Bab ini membahas tentang kesimpulan dari penelitian yang dilakukan mengenai peramalan debit bendungan menggunakan metode *backpropagation* dan algoritme genetika serta saran yang dapat dikembangkan penelitian berikutnya.

7.1 Kesimpulan

Dari hasil penelitian peramalan debit bendungan menggunakan metode *backpropagation* dan algoritme genetika yang dilakukan dapat diperoleh kesimpulan sebagai berikut:

1. Metode *backpropagation* dan algoritme genetika dapat diimplementasikan untuk peramalan debit bendungan dengan cara melakukan pelatihan bobot jaringan saraf tiruan menggunakan algoritme genetika untuk mendapatkan nilai *fitness* terbaik dan menggunakan bobot terbaik hasil algoritme genetika untuk digunakan sebagai bobot awal dalam metode *backpropagation*
2. Berdasarkan hasil pengujian yang dilakukan parameter pelatihan optimal algoritme genetika adalah populasi=100, generasi=100, kombinasi Cr dan Mr berturut-turut 0,6 dan 0,4. Parameter pelatihan optimal metode *backpropagation* adalah jumlah iterasi=500, nilai *learning rate*= 0,7. Hasil pelatihan menggunakan parameter optimal mendapatkan nilai MSE = 0,03062.
3. Berdasarkan hasil pengujian yang dilakukan menggunakan parameter optimal terhadap data uji, nilai rata-rata MSE yang dihasilkan pada peramalan menggunakan metode *backpropagation* adalah MSE=0,04417 dan menggunakan algoritme genetika dan metode *backpopagation* MSE= 0,04188. Hasil tersebut menunjukkan bahwa pemanfaatan algoritme genetika untuk optimasi bobot jaringan saraf tiruan menghasilkan hasil peramalan debit bendungan yang lebih baik apabila dibandingkan dengan hanya menggunakan metode *backpropagation* saja.

7.2 Saran

1. Data yang digunakan pada penelitian ini hanya menggunakan parameter time series data debit bendungan, perlu dipertimbangkan penambahan beberapa parameter seperti data tinggi muka air dan curah hujan untuk meningkatkan hasil peramalan debit bendungan.
2. Menggunakan beberapa metode optimasi lain seperti *Particle Swarm Optimization* (PSO) untuk membandingkan hasil optimasi yang dilakukan dengan algoritme genetika

DAFTAR PUSTAKA

Adwandha, D. P. D. E. R. P. A., 2017. Prediksi Jumlah Pengangguran Terbuka di Indonesia menggunakan. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer,* Vol. 1(No. 4), pp. 341-351.

Anindita, A. P. & Laksono, P., 2016. *Dam Water Level Prediction System Utilizing Artificial Neural Network Back Propagation.* Surabaya, 2016 International Conference on ICT For Smart Society.

Esfandani, A. &. N., 2015. Predicting air pollution in Tehran: Genetic algorithm and back propagation neural network. *Journal of AI and Data Mining,* 4(2), pp. 49-54.

Fausset, L., 2008. *Fundamentals of Neural Networks: Architectures, Algorithms and Applications.* s.l.:Oxford: Addisen-Wesley Publishing Compane, Inc..

Haviluddin, A. R., 2015. *A Genetic-Based Backpropagation Neural Network for Forecasting in Time-Series Data.* s.l., International Conference on Science in Information Technology (ICSITech).

Jayaraj, V. & Raman, N., 2016. *A genetic Algorithm Optimized Multilayer Perceptron for Software Defect Prediction.* s.l., IJARSE.

Liu, Q. & Liu, M., 2016. *A Fault Prediction Method Based on Modified Genetic Algorithm Using BP Neural Network Algorithm.* Budapest, Hungary, IEEE International Conference on Systems, Man, and Cybernetics'.

Mahmudy, W. F., 2015. *Dasar-Dasar Algoritma Evolusi.* Malang: Program Teknologi Informasi dan Ilmuy Komputer (PTIIK) Universitas Brawijaya.

Perum Jasa Tirta I, 2014. *Manual Operasi dan Pemeliharaan Bendungan Sengguruh,* Malang: Perum Jasa Tirta I.

Sari, N. R., Mahmudy, W. F. & Wibawa, A. P., 2016. *Backpropagation* on Neural Network Method for Inflation *Rate* Forecasting in Indonesia. *Int. J. Advance Soft Compu. Appl,* Vol 8(3).

Somlek, C., Kaewchainam, N., Simano, T. & So-In, C., 2016. *Usin backpropagarion neural network for flood forecasting in PhraNakon Si Ayutthaya, Thailand.* s.l., ICSEC 2015-19th international Computer Science and Engineering Conference.